80 x 24 Programmable Video Display Board

- User's guide -

copyright 1979 by: Gimix, Inc. Chicago, Illinois

CONTENTS:

Setting the board addresses
- MEMORY....2
- CONTROL PORTS....2

Using the board
- BLANKING....2
- MEMORY ENABLE/DISABLE....2
- MEMORY SELECTION....2
- DISPLAY MODES:
  - character type....3
  - character generators....3
  - display options....3
  - modes....3
  - programming the modes....4
- MODE SELECTION....4
- SCREEN MEMORY ACCESS....4
- SCROLLING....4
- THE CURSOR:
  - on/off....5
  - positioning....5
  - steady/flashing....5
- THE CHARACTER GENERATORS:
  - layout of characters in memory....5
  - programming the RAM character generator....5
- THE VERTICAL SYNC FEEDBACK....6
- THE 1-BIT LATCHED OUTPUT....6

DIAGRAMS:
- DIP switch setting....2
- Control ports (diagram 1)....7
- Memory layout (diagram 2)....8
- Mode programming worksheet....9
- Component layout....10
- Schematic....11
- Parts list....12

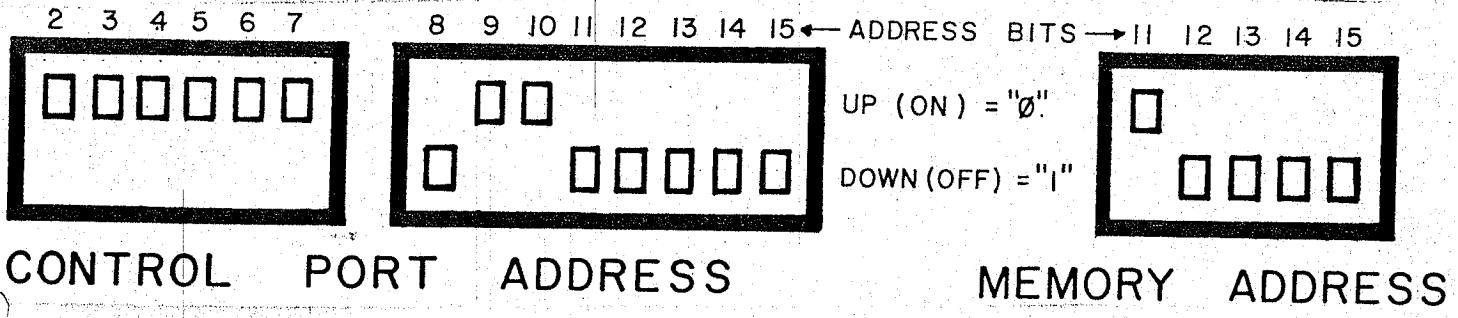
--SETTING THE BOARD ADDRESSES--

MEMORY:

The board uses a 2K block of CPU address space, starting at any 2K boundary. The GIMIX standard is F000 hex. Refer to the figure below to set the DIP switches.

CONTROL PORTS:

The board uses four consecutve bytes, starting at any four-byte boundary. The GIMIX standard setting is F900 hex. The bytes will be referred to as PORT 0 through PORT 3, with PORT 0 having the lowest address. Generally, the ports should not overlap the 2K memory space as this would disable the four memory locations overlapped. Refer to the figure below to set the DIP switches.



--USING THE BOARD--

BLANKING:

Video image blanking is controlled by PORT 0 BIT 4. When this bit is set to 0, as it is upon Reset, the video image will be blanked. This bit must be set to 1 in order to get a picture on the screen.

MEMORY ENABLE:

Memory enable is controlled by PORT 0 BIT 0. When this bit is set to 0, as it is upon Reset, the RAM on the board will NOT appear on the CPU bus. This bit must be set to 1 for the RAM to appear on the CPU bus.

MEMORY SELECTION:

There are two independent 2K blocks of RAM on the board. Which of these two blocks appears in the 2K memory address space is determined by PORT 0 BIT 1. When this bit is set to 0, as it is upon Reset, the SCREEN MEMORY will appear on the CPU bus. When this bit is set to 1, the CHARACTER GENERATOR MEMORY will appear on the CPU bus.

MODE PROGRAMMING:

Character types -

The ASCII characters can be divided into four GROUPS of 32 characters. These GROUPS are differentiated by the upper two bits (6 and 5) of the character codes in them. Bit 7, while not used in the ASCII code, is used by the board to determine which SLOT a character is in:

bits: 7 6 5	SLOT	character GROUP
0 0 0	SLOT 0	control characters
0 0 1	SLOT 0	numerals
0 1 0	SLOT 0	upper case alphabet
0 1 1	SLOT 0	lower case alphabet
1 0 0	SLOT 1	control characters
1 0 1	SLOT 1	numerals
1 1 0	SLOT 1	upper case alphabet
1 1 1	SLOT 1	lower case alphabet

Character generators -

There are three on-board character generators, each containing patterns for 128 characters. A fourth pseudo-generator produces only blank spaces. A two-bit code is used to reference the character generators:

code	generator
00	ROM A
01	ROM B
10	RAM
11	blank

Display Options -

A character may be displayed on the screen with one of the following options, represented by a two-bit code:

code	display option
00	NORMAL
01	HALF INTENSITY
10	INVERSE VIDEO
11	HALF INTENSITY + INVERSE VIDEO

Modes -

A mode is a set of commands that tells the board how to display characters contained in the screen memory. The board is capable of storing two modes and operating with either one. They are referred to as MODE 0 and MODE 1, and are independently programmable. The mode being programmed does not depend on which mode the board is using at the time.

Programming -

The function of PORT 1 is selected by PORT 0 BIT 2. When this bit is set to 1, PORT 1 becomes a write-only mode pprogramming register.

To completely program one display mode, eight bytes must be outputted to PORT 1. Each byte will program one of the eight character types for that mode. To do that each byte must contain the following information telling the board exactly what is being programmed:

which MODE? (1 bit)
which SLOT in that MODE? (1 bit)
which CHARACTER GROUP in that SLOT? (2 bits)

Each byte must also contain the following information telling the board WHAT TO USE for THAT CHARACTER GROUP in THAT SLOT of THAT MODE:

which DISPLAY OPTION? (2 bits)
which CHARACTER GENERATOR? (2 bits)

The placement of this information within the byte is shown explicitly in diagram 1. It is not necessary that all eight bytes be written when programming a mode. Once any particular character group within a slot of either mode has been programmed, it will stay that way until it is programmed differently.

MODE SELECTION:

The mode that is currently beig used for display is determined by PORT 0 BIT 3. When this bit is set to 0, as it is upon Reset, MODE 0 is used for display. When this bit is set to 1, MODE 1 will be used.

SCREEN MEMORY ACCESS:

The lower 1920 bytes of the screen memory represent the character spaces on the screen. They are partitioned into 24 successive groups. The first (lowest) group is called LINE 0, and the last (highest) is LINE 23. Each group is 80 bytes in length and represents one line on the screen. The first byte of each group is the leftmost character of that line. The highest 128 bytes of the screen memory are not used, but are accessible by the CPU. Refer to diagram 2.

SCROLLING:

When PORT 0 BIT 2 is set to 0, as will be upon Reset, PORT 1 becomes a read/write register holding the line number (0-23) of the line at the BOTTOM of the screen. The rest of the lines follow in numerical order starting at the top of the screen.

This register must be written to once after power-up before its contents will be valid for reading. Do not write a value greater than the highest line number (23) as this will disrupt the screen image -- a situation which can be corrected by writing a valid number back to the register.

THE CURSOR:

On/Off -

The cursor is controlling by PORT 0 BIT 5. When this bit is set to 0, as it is upon Reset, the cursor is turned off. When this bit is set to 1 the cursor is on.

Positioning -

The cursor inverts the video output for the position (0-79) denoted by the contents of PORT 2, on the line (0-23) denoted by the contents of PORT 3.

Flashing -

The cursor may be flashing or steady, as determined by PORT 0 BIT 6. When this bit is set to 0, as it is upon Reset, the cursor will be steady. When this bit is set to 1, the cursor will flash. The flashing is about a 25% duty cycle, so that the normal status of the character behind the cursor is determinable.

CHARACTER SET PROGRAMMING

Layout of character generators -

Each character generator is 2K bytes of memory. This memory is divided into 128 segment, each containing a pattern for one character. These segments are numbered 0-127 (0 representing the segment starting at the lowest memory address), corresponding to the ASCII code.

Each segment is 16 bytes long, sufficient to store an 8 x 16 character pattern. For the standard 80 x 24 version of the monitor, the character grid size is 8 x 10, so the last 6 bytes of each segment are not used. The top row of a character grid is represented by the first byte (lowest memory address) of a segment. The leftmost column is represented by BIT 7, and the rightmost by BIT 0. This arrangement is depicted is diagram 2.

Programming the RAM character generator -

To program the character generator, PORT 0 BIT 1 must be set to 1. This puts the character generator memory on the CPU bus.

THE VERTICAL SYNC FEEDBACK:

The video board refreshes the video image 60 times a second. If any of the basic refresh parameters (scroll value, mode program, or blanking) is changed during a refresh, part of the refresh is done with one set of parameters, and part with a different set. This produces a highly visible and annoying disruption of the video image. To allow this disruption to be avoided, the VERTICAL SYNC signal is available as PORT 0 BIT 7. At the end of each refresh scan, the video board generates this signal and PORT 0 BIT 7 reads as 1 for 190 usec. 1 msec after the signal goes away, the next refresh scan begins. During the 1.19 msec interval between refresh scans, the parameters may be altered and the whole screen will be refreshed with the same parameters, thus avoiding the disruption. The screen memory and the character generator memory can be accessed at this time without causing hash on the screen.

THE 1-BIT LATCHED OUTPUT

The VERTICAL SYNC FEEDBACK on PORT 0 BIT 7 is read-only. Data written to this bit apppears as a latched LS TTL output level. The signal is available at two pads: one has the true signal, the other has the inverse of the signal. This output may be used to control "beep" signals, drive a small speaker, flash an LED, etc.

CONTROL PORT BIT FUNCTIONS

PORT Ø

CONTROL FUNCTIONS

7	6	5	4	3	2	1	Ø

(INITIALIZED TO ALL ZEROES ON RESET)

VERTICAL SYNC
(READ ONLY)
1 = VERTICAL SYNC

CURSOR SELECT
Ø = STEADY
1 = FLASHING

CURSOR CONTROL
Ø = OFF
1 = ON

VIDEO BLANKING
Ø = BLANKED
1 = NORMAL DISPLAY

MEMORY ENABLE/DISABLE
Ø = DISABLED
1 = ENABLED

MEMORY SELECT
Ø = SCREEN MEMORY
1 = RAM CHARACTER GENERATOR

PORT 1 FUNCTION SELECT
Ø = SCROLL REGISTER
1 = MODE PROGRAMMING PORT

MODE SELECT
Ø = USE MODE Ø
1 = USE MODE 1

PORT 1

SCROLL REGISTER

7	6	5	4	3	2	1	Ø

(CONTENTS INVALID UNTIL WRITTEN TO)

LINE TO DISPLAY AT BOTTOM OF SCREEN (Ø–23)

MODE PROGRAMMING PORT (WRITE ONLY)

7	6	5	4	3	2	1	Ø

MODE TO PROGRAM
Ø = MODE Ø
1 = MODE 1

SLOT TO PROGRAM
Ø = SLOT Ø
1 = SLOT 1

GROUP TO PROGRAM

BIT 5	BIT 4	CHARACTERS
Ø	Ø	CONTROL
Ø	1	NUMERIC
1	Ø	UPPER CASE
1	1	LOWER CASE

CHARACTER GENERATOR

BIT 1	BIT Ø	USE:
Ø	Ø	ROM A
Ø	1	ROM B
1	Ø	RAM
1	1	BLANK SPACES

OPTIONS

BIT 3	BIT 2	DISPLAY IN:
Ø	Ø	NORMAL FORMAT
Ø	1	HALF INTENSITY
1	Ø	INVERSE VIDEO
1	1	BOTH

PORT 2

CURSOR X POSITION

7	6	5	4	3	2	1	Ø

CHARACTER POSITION OF CURSOR (Ø–79)

PORT 3

CURSOR Y POSITION

—	—	5	4	3	2	1	Ø

LINE NUMBER OF LINE CONTAINING CURSOR (Ø–23)

GIMIX INC.
1337 W. 37th PLACE, CHICAGO, IL 60609.

SCALE	DRAWN BY MOSES OKEKE
DATE 12–5–78	REVISED

80x24 VIDEO BOARD-CONTROL PORT MAP

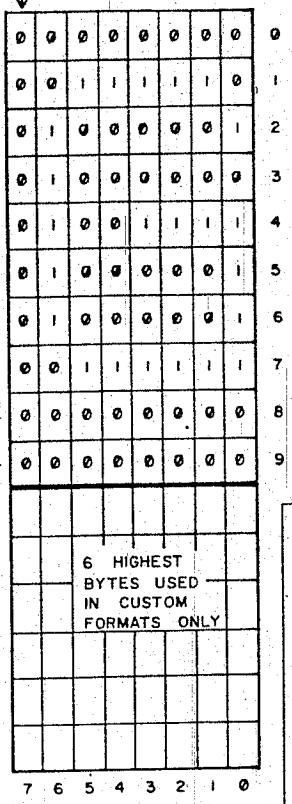
DRAWING NUMBER A 24 0038-1

LAYOUT OF CHARACTER GENERATORS

LEAVE THIS COLUMN BLANK FOR SPACE BETWEEN CHARACTERS

LEAVE THIS ROW BLANK FOR SPACE BETWEEN LINES

NOTE: THIS IS ONLY AN EXAMPLE OF ONE WAY TO USE THE CHARACTER GRID — ALL 80 PIXELS ARE USABLE.



THIS DIAGRAM REPRESENTS THE ACTUAL HEIGHT TO WIDTH RATIO (2:1) OF THE CHARACTER GRID AS IT WOULD APPEAR ON THE SCREEN.

ROW NUMBERS

THESE ROWS ARE USED FOR DESCENDING CHARACTERS.

NORMAL FORMAT:
0 = BLACK
1 = WHITE

INVERSE VIDEO:
0 = WHITE
1 = BLACK

HALF INTENSITY:
WHITE BECOMES GRAY
BLACK STAYS BLACK

BIT NUMBER

ONE CHARACTER

MEMORY ADDRESS OF THE BYTE REPRESENTING ANY PARTICULAR ROW IS EQUAL TO:

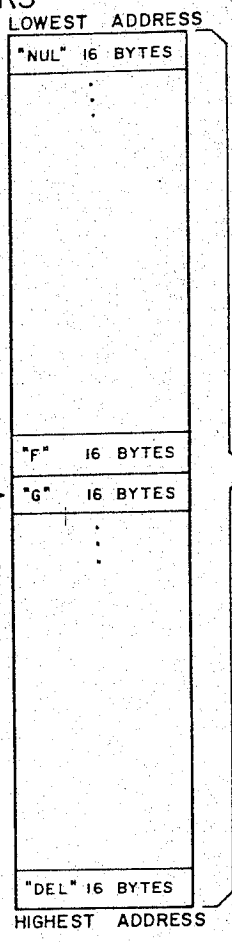
STARTING ADDRESS OF CHARACTER GENERATOR MEMORY

+

ASCII CODE OF CHARACTER TIMES 16 (DECIMAL)

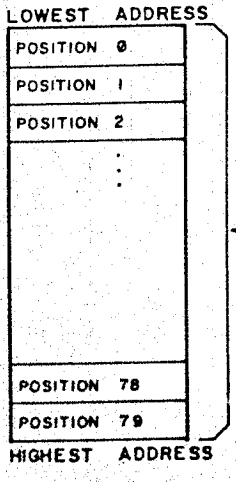
+

NUMBER OF THAT ROW (0-9)

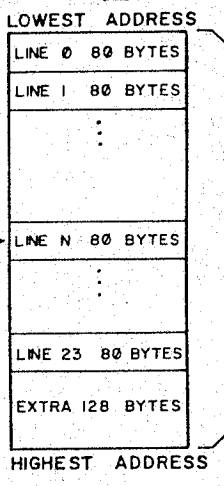


POSITION 0 1 2 ... G I M I X I N C 78 79

EITHER THE SCREEN MEMORY OR THE RAM CHARACTER GENERATOR MAY APPEAR IN THE BOARD'S 2K BYTE ADDRESS SPACE.



ONE LINE OF 80 CHARACTERS



LAYOUT OF SCREEN MEMORY

GIMIX INC.
1337 W. 37th PLACE, CHICAGO, IL 60609

SCALE	APPROVED BY	DRAWN BY MOSES C. OKE
DATE 12-6-78		REVISED

80 x 24 VIDEO BOARD — MEMORY MAP

DRAWING NUMBER A 240038-2

MODE PROGRAMMING WORKSHEET

MODE	MOST SIGNIFICANT BITS OF CHARACTERS IN SCREEN MEMORY: SLOT (BIT 7)	CHARACTER GROUP (BIT 6)	(BIT 5)	CHARACTER TYPE	INVERSE VIDEO ? Ø = NO 1 = YES	HALF INTENSITY? Ø = NO 1 = YES	WHICH CHARACTER GENERATOR ? ØØ = ROM A, 1Ø = RAM Ø1 = ROM B, 11 = BLANK		HEX VALUE TO WRITE TO PORT 1 *
Ø	Ø	Ø	Ø	SLOT Ø CONTROL					Ø
	Ø	Ø	1	SLOT Ø NUMERIC					1
	Ø	1	Ø	SLOT Ø UPPER CASE					2
	Ø	1	1	SLOT Ø LOWER CASE					3
	1	Ø	Ø	SLOT 1 CONTROL					4
	1	Ø	1	SLOT 1 NUMERIC					5
	1	1	Ø	SLOT 1 UPPER CASE					6
	1	1	1	SLOT 1 LOWER CASE					7
1	Ø	Ø	Ø	SLOT Ø CONTROL					8
	Ø	Ø	1	SLOT Ø NUMERIC					9
	Ø	1	Ø	SLOT Ø UPPER CASE					A
	Ø	1	1	SLOT Ø LOWER CASE					B
	1	Ø	Ø	SLOT 1 CONTROL					C
	1	Ø	1	SLOT 1 NUMERIC					D
	1	1	Ø	SLOT 1 UPPER CASE					E
	1	1	1	SLOT 1 LOWER CASE					F

LEFT HEX – DIGIT

RIGHT HEX – DIGIT

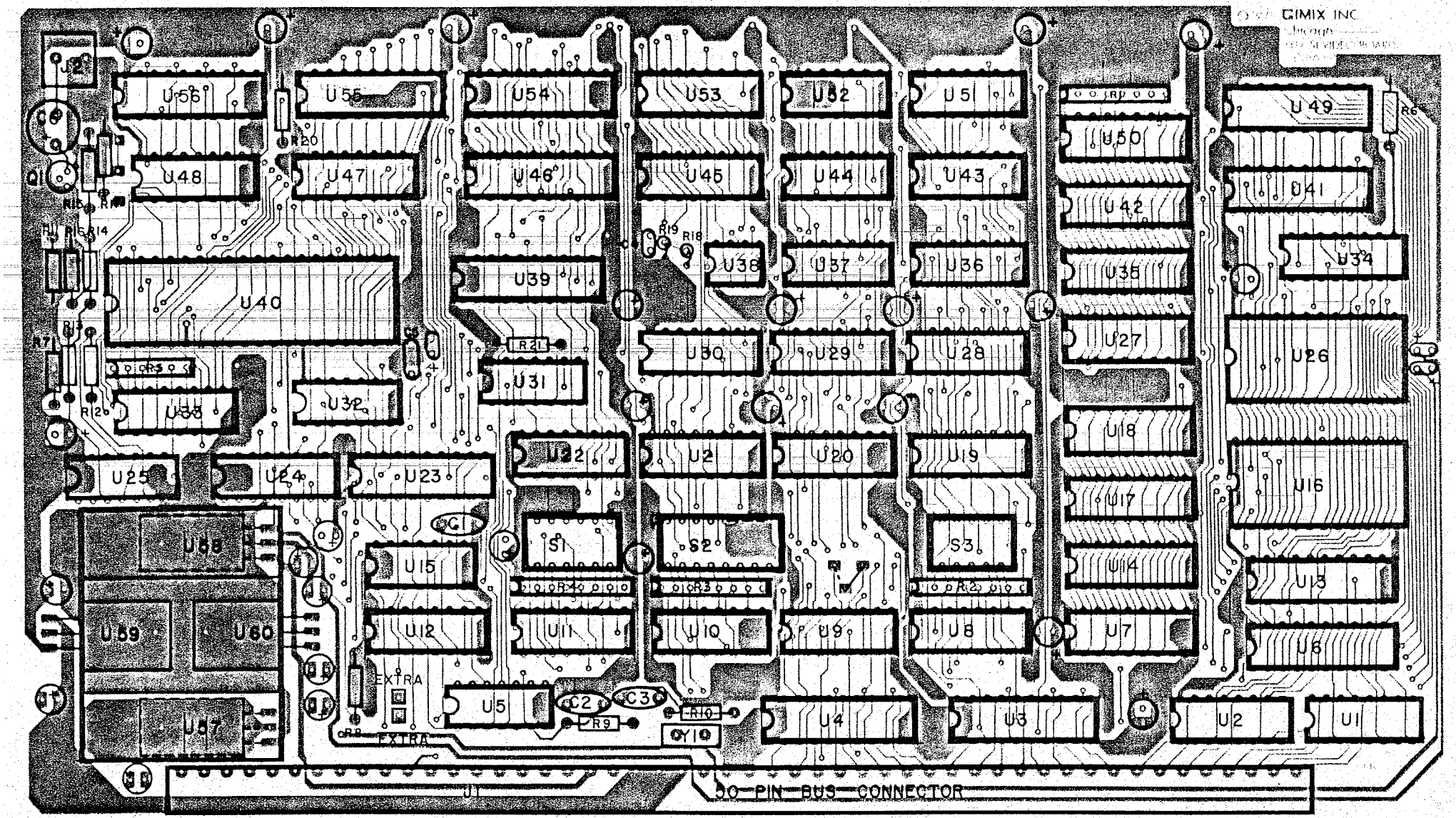
* PORT Ø BIT 2 MUST BE SET TO 1 IN ORDER FOR PORT 1 TO FUNCTION AS THE MODE PROGRAMMING PORT.

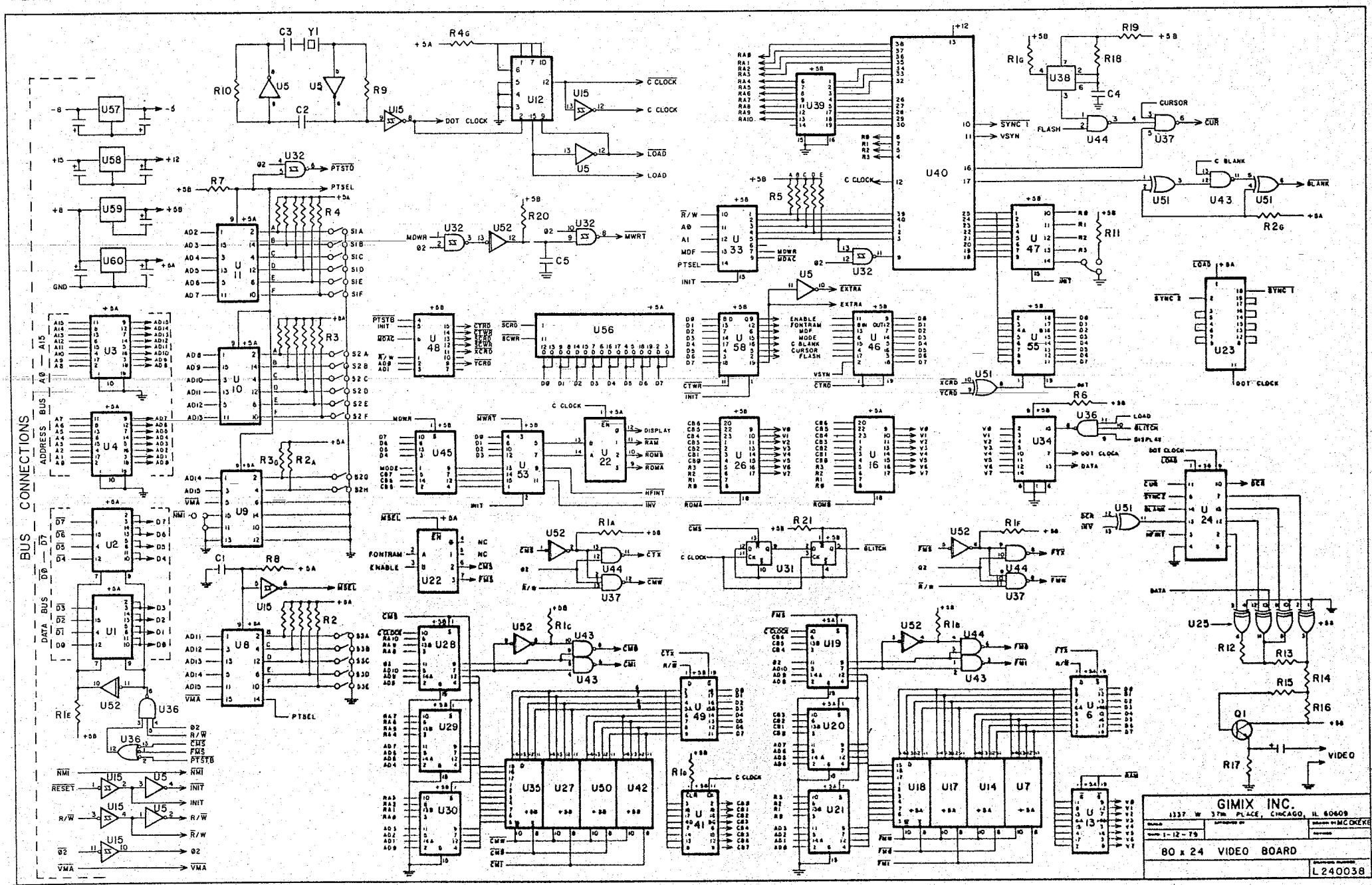
GIMIX INC.
1337 W. 37th PLACE, CHICAGO, IL 60609

SCALE
DATE 12 – 8 – 78
DRAWN BY MOSES C. OKEKE
REVISED

VIDEO BOARD – MODE WORKSHEET

DRAWING NUMBER A 240038-3


GIMIX INC
U56
U55
U54
U53
U52
U51
U49
U48
U47
U46
U45
U44
U43
U50
U41
U42
U40
U39
U38
U37
U36
U35
U34
U33
U32
U31
U30
U29
U28
U27
U26
U25
U24
U23
U22
U21
U20
U19
U18
U16
U17
U15
U14
U13
U12
U11
U10
U9
U8
U7
U6
U5
U4
U3
U2
U1
U58
U59
U60
U57
S1
S2
S3
C1
C2
C3
R9
R10
EXTRA
EXTRA
J1
50 PIN BUS CONNECTOR


BUS CONNECTIONS
ADDRESS BUS
DATA BUS
C3
Y1
R10
R9
C2
R4G
C CLOCK
DOT CLOCK
LOAD
R19
R18
R16
C4
CURSOR
FLASH
SYNC 1
VSYN
R7
R4
R5
R11
R20
C5
PTSTO
PTSEL
MDWR
MWRT
R3
R2
R3G
R2A
S2A
S2B
S2C
S2D
S2E
S2F
S2G
S2H
S3A
S3B
S3C
S3D
S3E
R1A
R1B
R1C
R1D
R1E
R1F
R1G
R21
R6
R12
R13
R14
R15
R17
Q1
VIDEO
C BLANK
BLANK
R26
SYNC 2
DOT CLOCK
GIMIX INC.
1337 W 37TH PLACE, CHICAGO, IL 60609
80 x 24 VIDEO BOARD
L240038

Parts list for Gimix 80 x 24 Video Board

QTY	PART	LOCATION
Integrated circuits		
2	8835	U1,2
4	74LS244	U3,4,13,46
1	74LS04	U5
3	74LS245	U6,49,55
8	2114	U7,14,17,18,27,35,42,50
4	8136	U8,9,10,11
1	74LS161	U12
1	74LS14	U15
2	TMS2716 **	U16,26
7	74LS157	U19,20,21,28,29,30,45
1	74LS139	U22
1	74LS377	U23
1	74LS378	U24
1	74LS136	U25
1	74LS74	U31
1	74LS132	U32
2	6331-1J or equiv. *	U33,47
1	74LS166	U34
2	74LS10	U36,37
1	555	U38
1	74S471 *	U39
1	CRT5027	U40
2	74LS273	U41,54
2	74LS00	U43,44
1	74LS138	U48
1	74LS86	U51
1	74S189	U53
1	74LS374	U56
1	7905	U57
1	7812	U58
2	7805	U59,60
1	74LS05	U52

* Parts marked with "*" are PROMs which must be programmed with correct Gimix data.

** U26 is the ROM A character generator (standard Gimix character set supplied). U16 is for the optional ROM B character generator (not supplied).

Resistors (1/4 watt - values in ohms)		
4	4.7K x 7 SIP	R1,2,3,4
1	4.7K x 5 SIP	R5
3	4.7K	R6,11,21
2	470	R7,8
2	330	R9,10
1	680	R12
2	180	R13,14
1	1.2K	R15
1	120	R16
1	100	R17
1	15K	R18
1	33K	R19
1	1.2K	R20

Capacitors

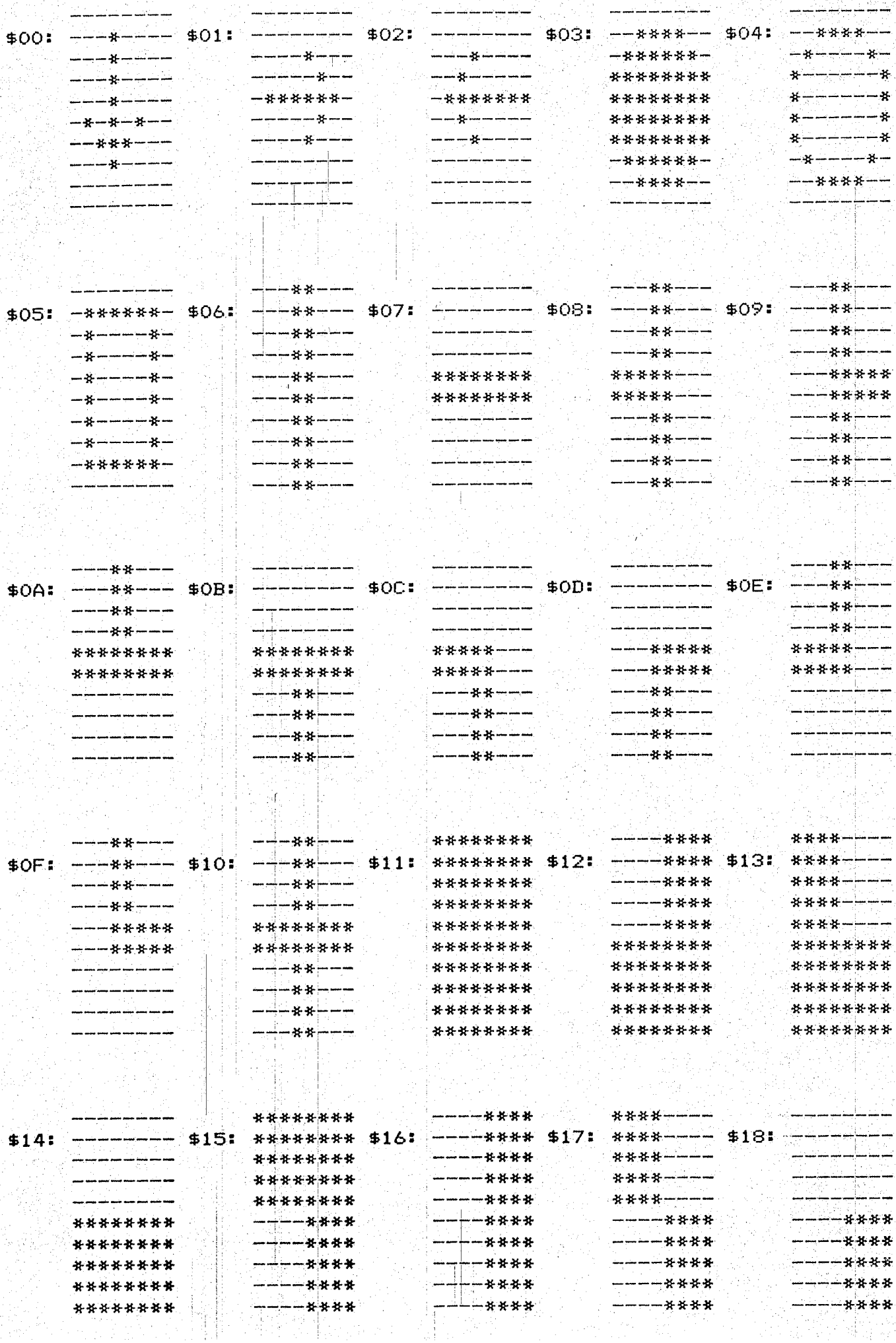
1	100 pF mica	C1
1	.001 uF Mylar	C2
1	33 pF mica	C3
1	10 uF 25V tantalum	C4
1	200 pF mica	C5
1	100 uF 16V electrolytic	C6
10	10 uF 25V tantalum	
20	.047 uF ceramic	

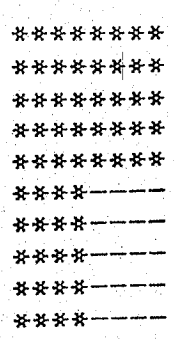
Other parts

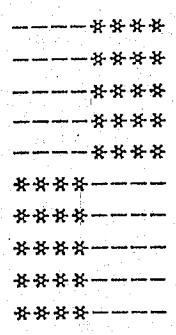
1	2N2222	Q1
1	15.0912 MHz crystal	Y1
1	50-pin bus connector	J1
1	2-position barrier block	J2
1	6-position DIP switch	SW1
1	8-position DIP switch	SW2
1	5-position DIP switch	SW3
2	THM 6071 heatsink top	
2	THM 6070 heatsink bottom	
2	silicone pads	
4	4-40 x 5/16 machine screws	
4	4-40 hex nuts	

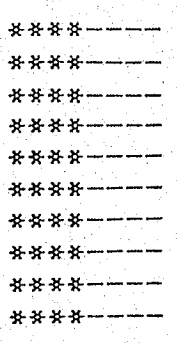
I.C. sockets

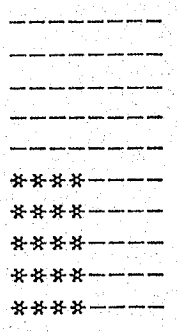
1	40-pin
2	24-pin
12	20-pin
8	18-pin
21	16-pin
11	14-pin
1	8-pin
1	index key for J1
1	plastic crystal spacer
1	P.C. board #24-0038

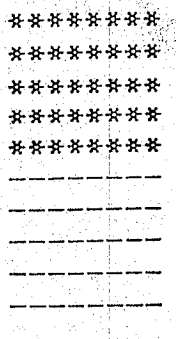

$00:
$01:
$02:
$03:
$04:
$05:
$06:
$07:
$08:
$09:
$0A:
$0B:
$0C:
$0D:
$0E:
$0F:
$10:
$11:
$12:
$13:
$14:
$15:
$16:
$17:
$18:

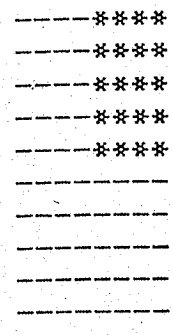
$19:


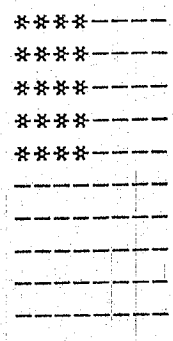
$1A:


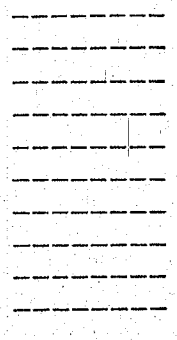
$1B:


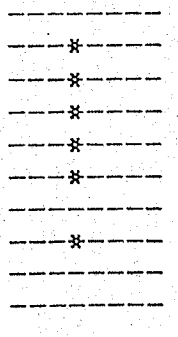
$1C:


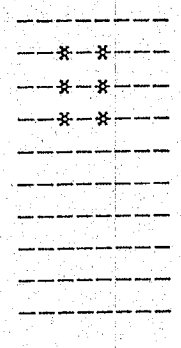
$1D:


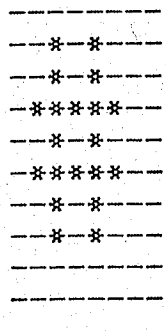
$1E:


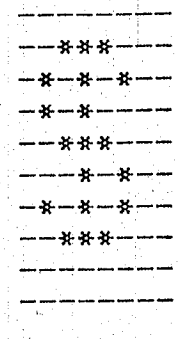
$1F:


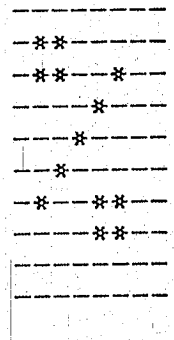
[ ]
$20:


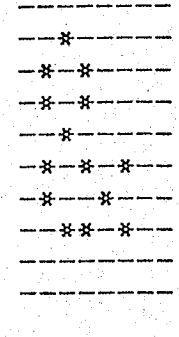
[!]
$21:


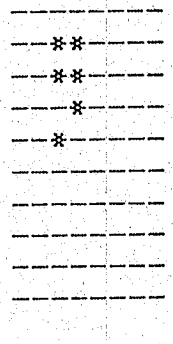
["]
$22:


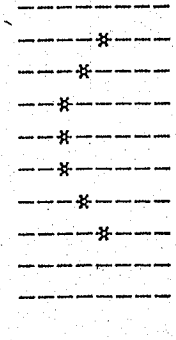
[#]
$23:


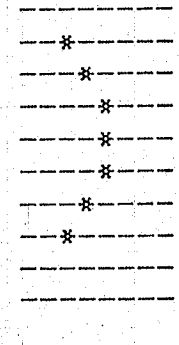
[$]
$24:


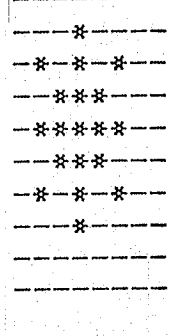
[%]
$25:


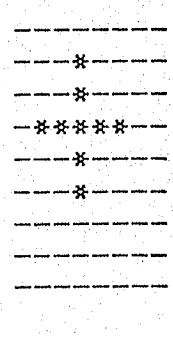
[&]
$26:


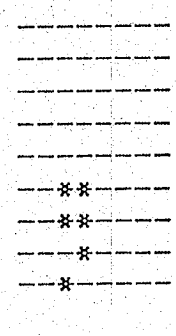
[']
$27:


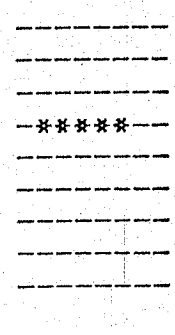
[(]
$28:


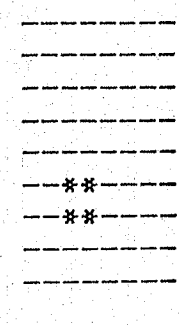
[)]
$29:


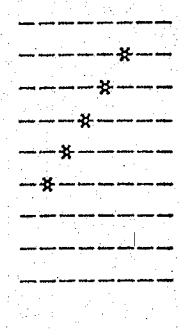
[*]
$2A:


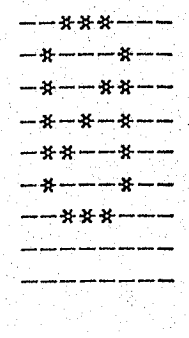
[+]
$2B:


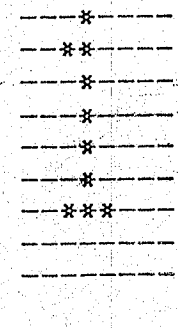
[,]
$2C:


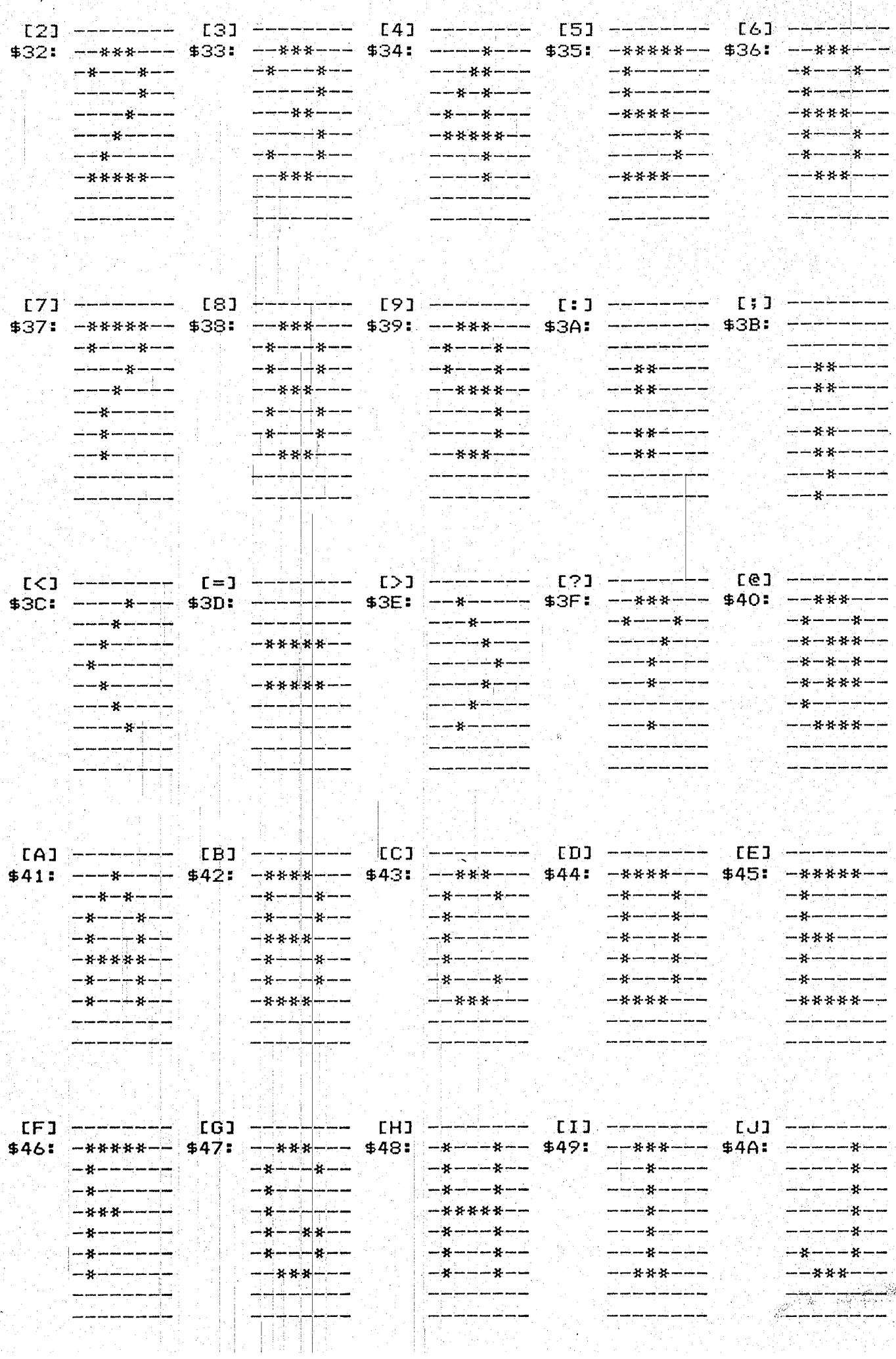
[-]
$2D:


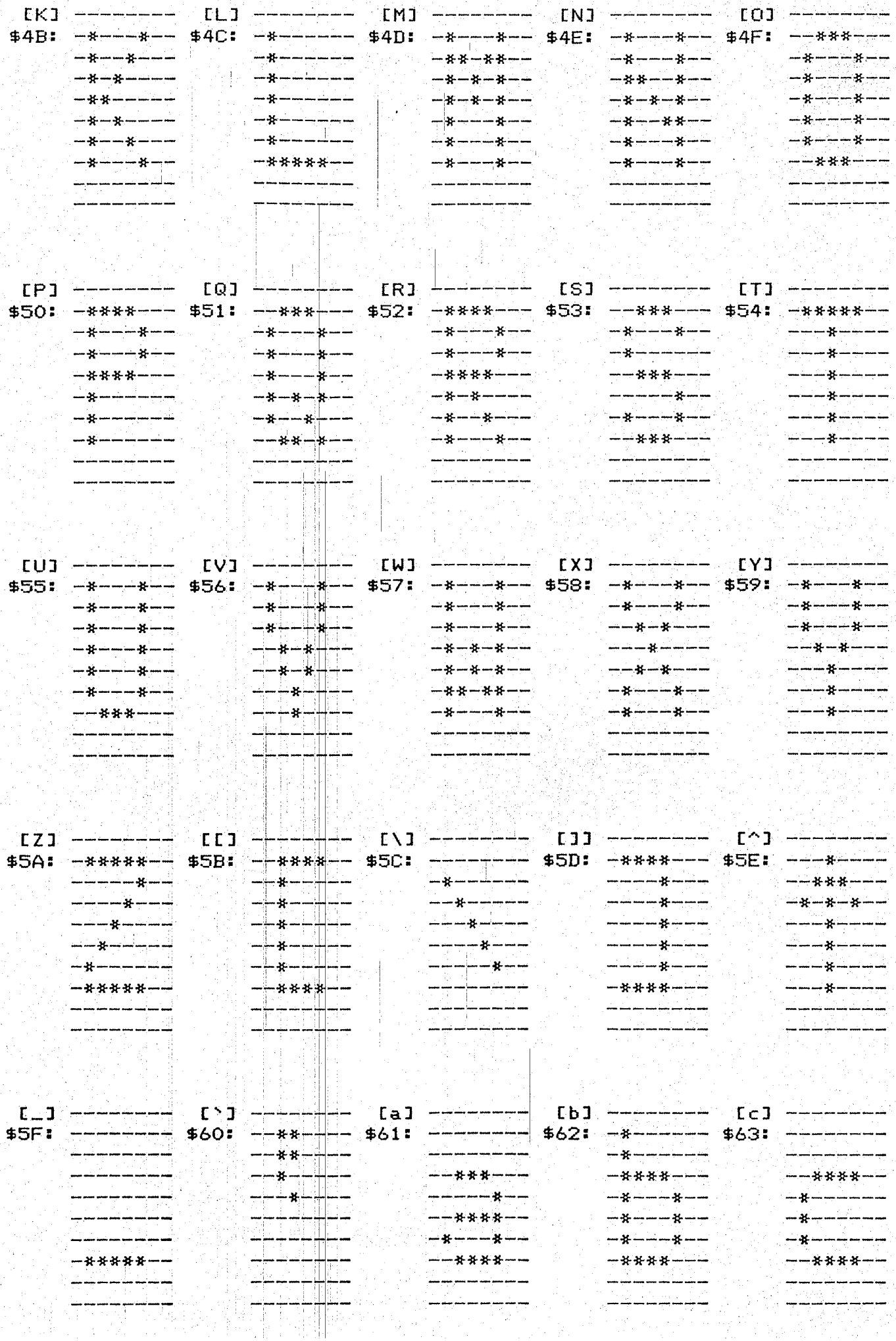
[.]
$2E:


[/]
$2F:


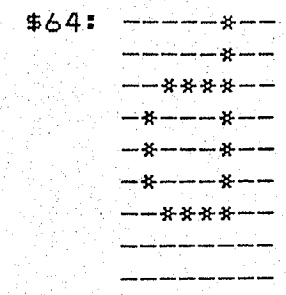
[0]
$30:


[1]
$31:


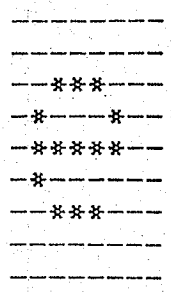

[2] $32:
[3] $33:
[4] $34:
[5] $35:
[6] $36:
[7] $37:
[8] $38:
[9] $39:
[:] $3A:
[;] $3B:
[<] $3C:
[=] $3D:
[>] $3E:
[?] $3F:
[@] $40:
[A] $41:
[B] $42:
[C] $43:
[D] $44:
[E] $45:
[F] $46:
[G] $47:
[H] $48:
[I] $49:
[J] $4A:


[K] $4B:
[L] $4C:
[M] $4D:
[N] $4E:
[O] $4F:
[P] $50:
[Q] $51:
[R] $52:
[S] $53:
[T] $54:
[U] $55:
[V] $56:
[W] $57:
[X] $58:
[Y] $59:
[Z] $5A:
[[] $5B:
[\] $5C:
[]] $5D:
[^] $5E:
[_] $5F:
[`] $60:
[a] $61:
[b] $62:
[c] $63:

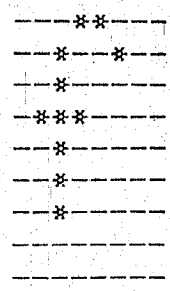
[d] $64:



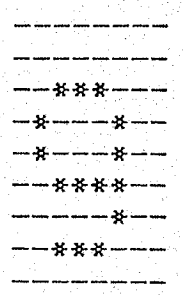
[e] $65:



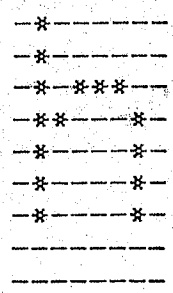
[f] $66:



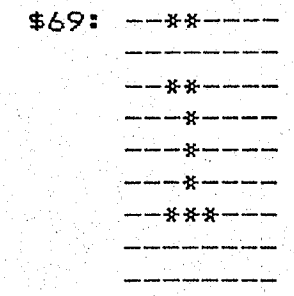
[g] $67:



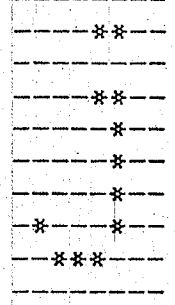
[h] $68:



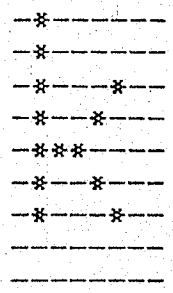
[i] $69:



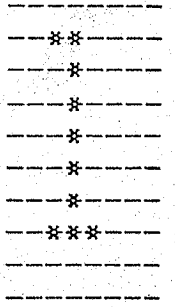
[j] $6A:



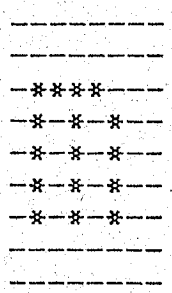
[k] $6B:



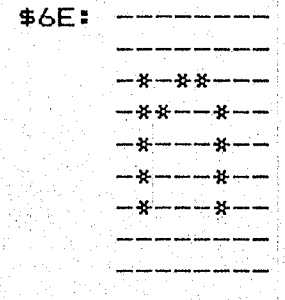
[l] $6C:



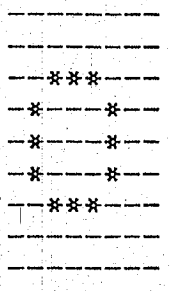
[m] $6D:



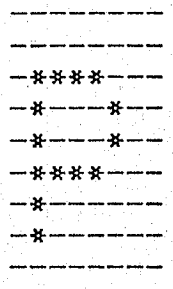
[n] $6E:



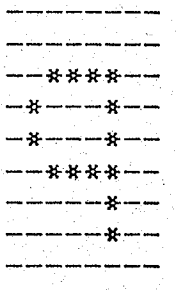
[o] $6F:



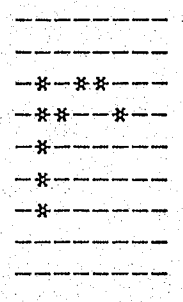
[p] $70:



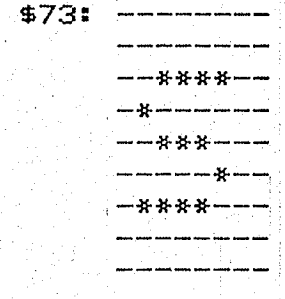
[q] $71:



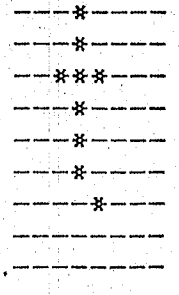
[r] $72:



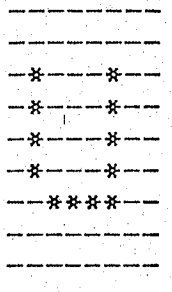
[s] $73:



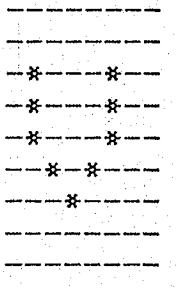
[t] $74:



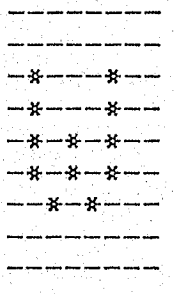
[u] $75:



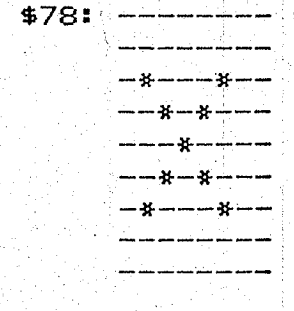
[v] $76:



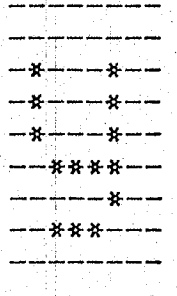
[w] $77:



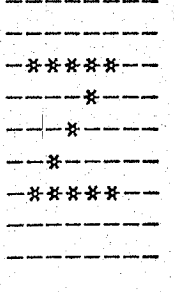
[x] $78:



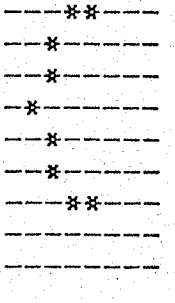
[y] $79:



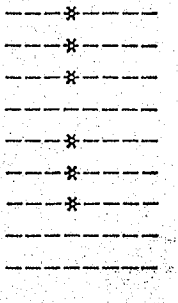
[z] $7A:

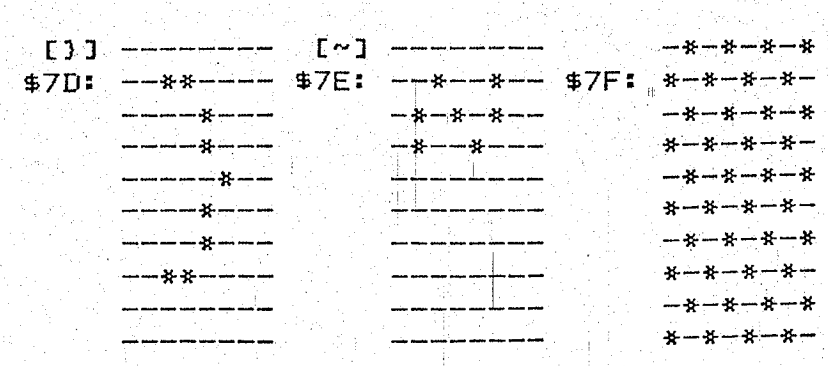


[{] $7B:



[|] $7C:




[}]
$7D:
[~]
$7E:
$7F:

Software Examples for the GIMIX 80x24 Video Board

by
Frank Alviani

Altho the GIMIX 80x24 video board is extremely powerful and flexible, this can be a drawback when it comes to explaining its operation - there is a great deal of information to absorb, much of it dealing with concepts not familiar to the average user. Altho there is a complete explanation given in the hardware manual provided with the board, GIMIX has been asked several times to give more examples of its practical use. This short manual will attempt to do just that.

This manual will explain the use of the GIMIX 80x24 video board in TSC BASIC in several configurations: (1) Using GMXBUG 3.0 and its video driver package, and (2) using the video board directly with PEEK and POKE statements, for anybody in the world using any monitor. The drivers explained can be included directly in any BASIC program for immediate use.

We will start with an explanation of the video board itself (more software oriented than the hardware manual) and then go into individual routines to accomplish specific functions. At the back of the manual will be listings of the complete support packages, ready for use as is. You can simply type in a support package once, and copy it into any BASIC programs (using the editor) you write in the future and be ready to go.

We will also cover the use of the video board by the assembler programmer in the last section of the manual. Rather than give an entire subroutine package, we will include examples inline which can be built into a subroutine utility package if desired by the user, or can be simply included inline in the coding if more efficient.

Video Board Facts of Life

The most important fact to keep in mind about the video board is that it is PROGRAMMABLE. Remember this at all times! Unlike a TTY or the average terminal, you can alter the way in which it will display a given code (say $41 - ASCII "A") at any time under the control of the program itself, along with certain other attributes of the display. Changing the character display handling is done by you in the program itself - not by moving switches or soldering jumpers. Much of the flexibility of the video board - and much of the initial difficulty - comes from this ability to determine how a byte is displayed, on the fly.

Programming the GIMIX 80 x 24 video board can be broken down into 3 major phases. The first phase is the actual programming of the 5027 video controller chip. This chip has several internal registers that determine such things as the number of characters per line, the number of lines per screen, and the size of each character cell. These registers are not accessible by the CPU but are programmed automatically whenever the system is reset. The values stored in these registers are fixed in a PROM on the video board. Users who have need of a custom screen format should contact GIMIX for more information.

The second phase is the programming of the control ports. There are 4 control ports accessible by the user. Their functions are as follows:

Port 0 - basic functions of the board
Port 1 - mode port / scroll port
Port 2 - cursor X-position
Port 3 - cursor Y-position

Port 0 is reset to all zeros following a system reset and must be programmed by the user. Normally this port will be loaded by the system initialization routine with an initial value and the individual bits changed as required during execution of the main program. for example, the initialization routine might load the value 00110101 ($35) into port 0. This bit pattern enables the video board memory, selects the screen memory to be accesible on the bus, enables port 1 as the MODE programming port, unblanks the display, and sets the cursor on and steady. Please see the hardware manual for a detailed description of the function of each bit.

Port 1 serves 1 of 2 functions depending on how bit 2 of port 0 is set. If port 0/bit 2 = 0 then port 1 serves as the scroll register and contains the number of the line that will be displayed as the last line on the screen. If port 0/bit 2 = 1 then port 1 functions as the mode programming port. The programming of the mode port determines how the characters will be displayed on the screen. Both the mode and scroll ports must be set by the system initialization routine for proper operation.

Typically the scroll port would be loaded with $17 which sets line 23 to be the last line on the screen. The values to be loaded into the mode port can be determined using the worksheet, as we will show a little later.

Ports 2 and 3 are the X and Y cursor position ports. The values stored in these ports determine where on the screen the cursor will be displayed if it is enabled. Port 3 determines on which line the cursor will appear, and port 2 determines where on that line it will appear. Since the value of the scroll resister affects the order in which the lines are displayed on the screen its value must be considered when determining what to load into port 3 to select a particular line for the cursor. Normally the initialization routine sets the scroll port to line 23 and both cursor ports to 0, placing the cursor in the upper left corner of the screen.

Since the cursor is under software control the routines that write to the screen are responsible for updating the values of the cursor ports as characters are entered or removed from the screen memory.

Mode Programming

To program the mode port is to specify how each subset of the possible 256 displayable characters is to be displayed on the screen. There are 8 possible subsets that can be referred to. A set of 8 display specifications (describing exactly how each subset is to be displayed) is called a MODE. The video board can hold 2 sets of these specifications and thus operate in 1 of 2 MODES at any time. You can change which MODE is in use at any time under program control.

When you specify how 1 subset of the 256 characters available is to be displayed, you specify: which of 4 character generators will be used, whether normal or inverse video will be displayed, and whether normal or half intensity is used. Let's go into more detail now on how we divide up the 256 possible characters into subsets.

The first big division pivots on whether the high-order bit is 1 or 0. We call each division a SLOT; if the high-order bit is 0, we are referring to SLOT 0 - if its a 1, we're referring to SLOT 1. This will be important when we go and fill out a worksheet (in just a minute).

Within a SLOT, the ASCII characters naturally fall into 4 groups:

$00-$1F	Control characters
$20-$3F	Numeric characters
$40-$5F	Upper-case alphabetic characters
$60-$7F	Lower-case alphabetic characters

If you look at a 6800 reference card the groupings will become immediately obvious. We've included a copy below:

ASCII CODE BITS:	CONTROL		NUMERIC		UPPER CASE		LOWER CASE	
6,5,4→	000	001	010	011	100	101	110	111
↓3,2,1,0								
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	¦
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	≈
1111	SI	US	/	?	O	—	o	DEL

Each of the 4 groups within a SLOT can be seperately assigned a character generator, normal/inverse video, and normal/half intensity. A MODE thus consists of:

2 SLOTs each containing
4 GROUPs of characters (A total of 8 subsets)

Why do you need 2 modes? A possibility: your text editor normally has all characters using the standard character generator, normal intensity, normal video. But, if you decide to make a change, it simply switches modes - nothing else - so that all upper-case letters are normal intensity, but everything else is half-intensity. You don't need to search memory and flag all upper-case letters - just change 1 bit in 1 port to switch modes!

In order to achieve the arrangement above, we need to program the modes when the editor sets things up, and in order to program program the modes, we need to know what to program them with - which is why a worksheet is attached to the hardware manual. You can determine on a copy of the worksheet exactly what must be put into the mode port in order to get any desired result. We have filled out the attached worksheet as follows:

Normal operation - Mode 0 (Lines 1 - 8)
  Normal display, normal intensity, and usng ROM A in all slots
Update operation - Mode 1 (Lines 9 - 16)
  Slot 0 Control - Normal display, half intensity, ROM A
  Slot 0 Numeric - the same
  Slot 0 Upper case - Normal display, normal intensity, ROM A
  Slot 0 Lower case - Normal display, half intensity, ROM A
  Slot 1 - same as slot 0

Thus, if the byte $A0 is written to the mode port, slot 0 upper case characters in mode 1 will be displayed normally. If they had been displayed in some other fashion, they would change appearance immediately, but nothing else on the screen would be affected. If $F4 is then written to the mode port, all others remain unchanged, but slot 1 lower case letters in mode 1 will be displayed in half intensity.

We have attached a filled-out worksheet for just the situation described above - if you look at it you'll see just how one is filled out. The blank worksheet is attached to the hardware manual to allow you to copy it for later use.

Altho a mode consists of a set of 8 specifications, it is not necessary to change all 8 each time a change is made. Any of the 16 possible specifications that can be store can be updated independently - you can change 1, 2 or all of them at any time.

Additional Features

Other features of the video board allow you to position the cursor at any location on the screen (in X,Y coordinate form), select blinking or steady cursor, etc. We will cover the simpler features (like X-Y cursor positioning) first and then progress to the board programming features.

The escape processor is a feature of GMXBUG 3.0 that allows you to affect the video board by "printing" strings of characters that are preceeded by an escape character ($1B). When an escape character is detected by GMXBUG, it switches from its normal "display" mode (put the incoming character into the video refresh memory) to "command" mode (interpret the following bytes as commands to change the operation of the video board). After the escape character comes an "op code" that tells GMXBUG what function to perform, and data bytes. There can be either a fixed number of these data bytes, or a byte count may preceed them. Once the function has been performed by GMXBUG, it reverts to its normal "display" mode.

When the system is turned on, if you have GMXBUG 3.0 installed, all the ports and modes are automatically initialized to "normal" values. You can then change them as needed, but need not worry about starting up. If you are using some other monitor or system , you will have to write an initialization routine yourself. This is not very difficult.

Video Support Routines in BASIC

All of the following routines, until further notice, apply to both the user with GMXBUG 3.0 and the user without it. The exact coding will differ (GMXBUG uses the escape processor in the output module, while the other version POKES directly to the video board ports) but the effect is the same. All coding is in TSC BASIC, which has a HEX function to save the hex -> decimal conversion effort on the part of the programmer.

Move Cursor Subroutine 9050

Sample Call: 200 X9=30:Y9=12:GOSUB 9050

This routine places the cursor at LINE Y9 in COLUMN X9. In the sample call, the cursor will end up at line 12, column 30. If a PRINT statement is executed at this point, that is where the output will start.

Blinking Cursor Subroutine 9140

Sample Call: 200 GOSUB 9140

This subroutine sets the cursor to blinking mode (about 25 per cent duty cycle) at the current location.

Steady Cursor Subroutine 9150

Sample Call: 200 GOSUB 9150

This subroutine sets the cursor to steady mode at the current location.

Send Output to Printer Subroutine 9180 (GMXBUG only)

Sample Call: 200 GOSUB 9180

This subroutine uses the output-device feature of GMXBUG 3.0 . All output is sent to the hardcopy device thru the GMXBUG printer routine till further notice. Nothing will go to the screen. Control characters will be transmitted, but printer response depends on the device itself.

Send Output to Screen Subroutine 9160 (GMXBUG only)

Sample Call: 200 GOSUB 9160

This subroutine is the converse of subroutine 9180 described above. All output is sent to the video board, which is the normal state of affairs.

Send Output to both Screen and Printer Subroutine 9170 (GMXBUG only)

Sample Call: 200 GOSUB 9170

This routine sends the output to both devices until further notice.

Output a Tone Subroutine 9210 (GMXBUG only)

Sample Call: 200 D9=25:P9=55:GOSUB 9210:PRINT CHR$(7);

This assumes you have a speaker attached to the toggleable 1-bit port of the video board (see the hardware manual). GMXBUG has the facilities for setting the "pitch" of the tone (by setting the period between toggles) and the length of the tone (by setting the duration). See the GMXBUG 3.0 manual for details. This sets the pitch and length of the tone produced by outputting a "bell" character ($07).

Mode Control Subroutines

The following subroutines are used to allow you to program 1 character subset (group), an entire mode, switch "slots", etc.

Select a Mode Subroutines 9060 / 9070

```
Sample Call:  200  GOSUB 9060   (Select mode 0)
              210  GOSUB 9070   (select mode 1)
```

These subroutines cause the video board to use the indicated set of group specifications (called "modes"). Since a group of characters can be displayed differently in the different modes, you can change how a group of characters - the upper case letters, for example - are displayed simply by changing modes, and changing nothing else.

Switch to Slot 0 Subroutine 9100 (GMXBUG only)

Sample Call: 200 GOSUB 9100

This makes sure that all characters output to the video board from now on have the high-order bit (bit 7) set to 0 - all characters output are in slot 0.

Switch to Slot 1 Subroutine 9110 (GMXBUG only)

Sample Call: 200 GOSUB 9110

This subroutine ensures that all characters output from now on have the high-order bit (bit 7) set to 1 - all characters are in slot 1.

Program 1 Character Group Subroutine 9130

```
Sample Call:  200 M9(1)=HEX("28")
              210 GOSUB 9130
```

This subroutine changes the way in which 1 character group is displayed (1 group within 1 slot in 1 mode). In the example above, in mode 0, slot 0, upper case characters are displayed in inverse video. All other display groups are left unaltered.

Program a Mode Subroutine 9120

```
Sample Call: 200 FOR I=1 TO 8
             210 M9(I)=X1(I)
             220 NEXT I
             230 GOSUB 9120
```

This subroutine changes the display characteristics of an entire mode. In the example above array X1 holds the definitions of each group (as worked out using the worksheet) in the new mode. To use subroutine 9120, the new definitions must be loaded into array M9 (in any order) and then call 9120.

Programmable Character Set Subroutines

Select Font RAM Subroutine 9080

Sample Call: 200 GOSUB 9080

This subroutine selects the RAM used to hold the programmable character set to be accessible to the program. Either the font RAM or the video refresh RAM are accessible, but not both at the same time. With the font RAM accessible, POKEing a byte in will change the pattern of dots to be displayed for a particular code.

Select the Video Refresh RAM Subroutine 9090

Sample Call: 200 GOSUB 9090

This is the reverse of subroutine 9080; it makes the video refresh RAM accessible. This is the normal state of affairs.

Build a Character in the Font RAM Subroutine 9190

```
Sample Call:  200 FOR I=1 TO 10
              210 F9(I)=X(I)
              220 NEXT I
              230 F8=33
              240 GOSUB 9190
```

This subroutine is used to change the pattern of dots displayed for a particular character. The code whose pattern is to be changed is in F8; the new pattern of dots is in the array F9. In the example above, the array X holds the new "definition" for code 33 (which is "!" in the normal ASCII character set). The parameters are set up and then the subroutine is called to actually make the change.

To determine the contents of F9, you must fill out a grid of 10 rows by 8 columns with a dot wherever desired. Each row will be 1 entry in the array - determine its value by using 1 for a dot, 0 for no-dot, and converting the binary number to decimal. If you are using TSC BASIC, you can use the HEX function to save some effort.

Video Support in Assembler

We will now cover the user who is working with the video board in assembler. We are not, however, going to assume that any of the escape processor routines are available to him; he's doing a package that will be monitor independent, and is writing the support modules for "stand-alone" usage. We are not going to include examples of ALL the possible things that can be done with the video board, but just enough to give the user a "feel" for how to approach the board in assembler.

Assumptions for Assembler Samples

In the examples to follow, a number of assumptions are made. They are as follows:

```
VRAM    EQU   $F000      ALL VIDEO RAM HERE
*
CNTRL1  EQU   $F900      BOARD 1 PORTS
SCROL1  EQU   CNTRL1+1
MODE1   EQU   SCROL1
XCURS1  EQU   CNTRL1+2
YCURS1  EQU   CNTRL1+3
*
CNTRL2  EQU   $F910      BOARD 2 PORTS
SCROL2  EQU   CNTRL2+1
MODE2   EQU   SCROL2
XCURS2  EQU   CNTRL2+2
YCURS2  EQU   CNTRL2+3
*
*       A-REG NORMALLY DESTROYED
*
```

Altho references are made to "board 1" and "board 2" ports, this does NOT mean that 2 boards are required. We will be looking at examples that can manipulate both single and multiple boards in a system.

Cursor Manipulation

We are not going to get into the details of bookkeeping that are involved with normal cursor movement: backspacing, carriage returns as the last character on the line, etc. These can be studied in the GMXBUG 3.0 manual if the user desires. What we are going to deal with are the registers on the video board that affect the cursor.

```
        LDA A CNTRL1    GET CURRENT CONTROL PORT
        AND A #%10111111  STEADY CURSOR - TURN OFF BIT 6
        STA A CNTRL1    UPDATE PORT
*
        LDA A CNTRL1    GET CURRENT PORT
        ORA A #%01000000  FLASHING CURSOR - TURN ON BIT 6
        STA A CNTRL1    UPDATE PORT
*
*       ....
*
        LDA A XPOS      GET NEW COLUMN POSITION
        CMP A #79
        BLE   SUB1      CONTINUE IF VALID
        RTS             LEAVE WITHOUT DOING ANYTHING IF BAD
SUB1    EQU   *
        STA A XCURS1    CURSOR WILL NOW MOVE ON SAME LINE
*
        LDA A YPOS      GET NEW ROW NUMBER
        CMP A #23
        BLE   SUB2      CONTINUE IF VALID
        RTS             LEAVE WITHOUT DOING ANYTHING IF BAD
SUB2    EQU   *
        STA A YCURS1    CURSOR WILL MOVE IN SAME COLUMN
*
*       ALWAYS TEST X AND Y CURSOR VALUES FOR OUT-OF BOUND
*       CONDITIONS, AS THE BOARD WILL BEHAVE PECULIARLY IF
*       FED BAD DATA....
*       X LIMIT = 79    Y LIMIT = 23
*
```

Scrolling the Screen with Hardware

It is possible to scroll the screen without rewriting the whole refresh memory by using the hardware features of the video board. Done this way, it is only necessary to blank one line rather than move 2K characters around - a much faster operation!

The scroll register tells the video board which line is to be displayed on the bottom of the screen - normally this port is set to 23 (line 0 at the top, line 23 at the bottom). If we now write 0 to this port, line 0 will appear at the bottom of the screen, below line 23, and line 1 will be at the top. If line 0 is blank, we will now have a blank line available to fill in.

To access the scroll port is a little more complicated than the other ports, such as the x-cursor port. You must first alter the control port to indicate whether the mode port or the scroll port is what you wish to access. Then you can go ahead as usual...

```
        LDA A CNTRL1    GET CONTROL PORT
        AND A #%11111011   WE WANT THE SCROLL PORT
        STA A CNTRL1
*
        LDA A SCROL1    GET CURRENT LINE
        INC A
        CMP A #24
        BLT   BLANK     STILL IN RANGE
        CLR A
BLANK   STA A SCROL1    SET NEW LINE # AT BOTTOM
*
        LDA B #80       LOAD BYTE COUNT
        JSR   MULTAB    COMPUTE OFFSET INTO REFRESH RAM
        LDX   #VRAM     POINT TO MEMORY
        JSR   ADDAB     ADD TO BASE ADDRESS
        LDA B #80       RELOAD BYTE COUNT
        LDA A #$20      BLANK..
LOOP1   STA A 0,X       CLEAR LINE 0
        INX
        DEC B
        BNE   LOOP1
*
*       USER-SUPPLIED ROUTINES:
*       MULTAB: THIS MULTIPLIES THE A AND B REGISTERS
*         RESULT IN A - B
*
*       ADDAB: THIS ADDS THE A - B PAIR TO THE X-REGISTER
*
```

Manipulating Multiple Video Boards

This is one of the more exotic possibilities for the video board. Since the refresh/character generator RAM can be either enabled or disabled, it is possible to have several boards all at the same address, and manipulate them using common routines. The visual appearance of a board whose memory is no longer on the bus is no different from one that is accessable; thus, the observer cannot tell whether "his" board can be accessed or not. This exercise is going to use a slightly different approach from the ones above.

```
VRAM    EQU   $F000
*
PORTS0 EQU    $F900       BOARD 1 PORTS
PORTS1 EQU    $F910       BOARD 2 PORTS
*
CONTRL EQU    0
SCROLL EQU    1
MODE    EQU   1
XCURSR EQU    2
YCURSR EQU    3
*
        JSR   PICK1       ENABLE BOARD 1 ONLY
        LDA A #0          SET FLASHING CURSOR
        JSR   CURSOR
        LDA A #30         SET X-CURSOR
        JSR   MOVEX
        LDA A #20         SET Y-CURSOR
        JSR   MOVEY
*       ....
        JSR   PICK0       ENABLE BOARD 0 ONLY
        LDA A #1          SET SOLID CURSOR
        JSR   CURSOR
        CLR A             SET X TO COLUMN 0
        JSR   MOVEX
        CLR A             SET Y TO ROW 0
        JSR   MOVEY
*       ...
*** SUBROUTINES ***
PICK0   EQU   *
        LDX   #PORTS1     POINT TO CONTROL REGISTER
        LDA A CONTRL,X
        AND A #$FE        DISABLE OTHER BOARD
        STA A CONTRL,X
        LDX   #PORTS0   POINT TO OWN CONTROL REGISTER
        LDA A CONTRL,X
        ORA A #$1         TURN ON OWN BOARD
        STA A CONTRL,X
        RTS
*
```

```
PICK1   EQU    *
        LDX    #PORTS0
        LDA A CONTRL,X
        AND A #$FE
        STA A CONTRL,X
        LDX    #PORTS1
        LDA A CONTRL,X
        ORA A #$1
        STA A CONTRL,X
        RTS
*
*       IN THE FOLLOWING ROUTINES, THE X-REG MUST
*       POINT TO THE CONTROL PORTS FOR A VIDEO BOARD
*       ON ENTRANCE....
*
CURSOR EQU    *
        TST A
        BNE    SOLID
        LDA A #$40
        ORA A CNTRL,X
        STA A CNTRL,X
        RTS
SOLID   LDA A #$BF
        AND A CNTRL,X
        STA A CNTRL,X
        RTS
*
MOVEX   STA A XCURSR,X
        RTS
*
MOVEY   STA A YCURSR,X
        RTS
```

As you can see the possibilities are enormous. This has been merely a set of samples, not exhaustive or definitive coding techniques. Have fun!!

MODE PROGRAMMING WORKSHEET

MODE	MOST SIGNIFICANT BITS OF CHARACTERS IN SCREEN MEMORY: SLOT (BIT 7)	CHARACTER GROUP (BIT 6)	(BIT 5)	CHARACTER TYPE	INVERSE VIDEO? Ø = NO 1 = YES	HALF INTENSITY? Ø = NO 1 = YES	WHICH CHARACTER GENERATOR? ØØ = ROM A, 1Ø = RAM Ø1 = ROM B, 11 = BLANK		HEX VALUE TO WRITE TO PORT 1 *
Ø	Ø	Ø	Ø	SLOT Ø CONTROL	Ø	Ø	Ø	Ø	Ø Ø
	Ø	Ø	1	SLOT Ø NUMERIC					1 Ø
	Ø	1	Ø	SLOT Ø UPPER CASE					2 Ø
	Ø	1	1	SLOT Ø LOWER CASE					3 Ø
	1	Ø	Ø	SLOT 1 CONTROL	1	Ø	1	Ø	4 Ø
	1	Ø	1	SLOT 1 NUMERIC					5 Ø
	1	1	Ø	SLOT 1 UPPER CASE					6 Ø
	1	1	1	SLOT 1 LOWER CASE					7 Ø
1	Ø	Ø	Ø	SLOT Ø CONTROL		1			8 4
	Ø	Ø	1	SLOT Ø NUMERIC		1			9 4
	Ø	1	Ø	SLOT Ø UPPER CASE		Ø			A Ø
	Ø	1	1	SLOT Ø LOWER CASE		1			B 4
	1	Ø	Ø	SLOT 1 CONTROL		1			C 4
	1	Ø	1	SLOT 1 NUMERIC		1			D 4
	1	1	Ø	SLOT 1 UPPER CASE		Ø			E Ø
	1	1	1	SLOT 1 LOWER CASE		1			F 4

LEFT HEX – DIGIT

RIGHT HEX – DIGIT

* PORT Ø BIT 2 MUST BE SET TO 1 IN ORDER FOR PORT 1 TO FUNCTION AS THE MODE PROGRAMMING PORT.

BASIC Examples Sample:
Mode 0 - Normal
Mode 1 - Inverse except Upper Case

GIMIX INC.
1337 W. 37th PLACE, CHICAGO, IL 60609

SCALE

DRAWN BY MOSES C. OKEKE

DATE 12-8-78

REVISED

VIDEO BOARD – MODE WORKSHEET

DRAWING NUMBER A 240038-3

```
8990 REM ******************************************************
8998 REM * SUPPORT PACKAGE #1 FOR GIMIX 80x24 VIDEO BOARD *
8999 REM * CONFIGURATION: GMXBUG 3.0                          *
9000 REM ******************************************************
9001 REM
9002 REM AVAILABLE ROUTINES:
9003 REM VARIABLE USAGE:
9004 REM X9=X COORDINATE    Y9=Y COORDINATE
9005 REM M9(8) PROGRAMMING OF 1 MODE
9006 REM Z9=COUNTER  Z8=FONT ADDRESS  F9(10)=CHAR DEF
9007 REM F8=CHARACTER CODE
9008 REM D9=DURATION OF TONE  P9=PERIOD OF TONE
9020 REM ROUTINES AVAILABLE:
9021 REM 9050-SET X,Y COORDINATES
9022 REM 9060-SELECT MODE 0
9023 REM 9070-SELECT MODE 1
9024 REM 9080-SELECT FONT RAM
9025 REM 9090-SELECT REFRESH RAM
9026 REM 9100-SWITCH TO SLOT 0
9027 REM 9110-SWITCH TO SLOT 1
9028 REM 9120-PROGRAM A MODE
9029 REM 9130-PROGRAM JUST 1 CHARACTER GROUP
9030 REM 9140-BLINKING CURSOR
9031 REM 9150-STEADY CURSOR
9032 REM 9160-OUTPUT TO SCREEN
9033 REM 9170-OUTPUT TO BOTH PRINTER AND SCREEN
9034 REM 9180-OUTPUT TO PRINTER ONLY
9035 REM 9190-BUILD A CHARACTER IN FONT RAM
9036 REM 9210-OUTPUT A TONE TO THE SPEAKER
9049 REM
9050 REM *** SET X-Y COORDINATES
9051 PRINT CHR$(27);CHR$(1);CHR$(X9);CHR$(27);CHR$(2);CHR$(Y9);
9052 RETURN
9060 REM *** SELECT MODE 0
9061 PRINT CHR$(27);CHR$(7);
9062 RETURN
9070 REM *** SELECT MODE 1
9071 PRINT CHR$(27);CHR$(8);
9072 RETURN
9080 REM *** SELECT FONT RAM
9081 PRINT CHR$(27);CHR$(6);
9082 RETURN
9090 REM *** SELECT REFRESH RAM
9091 PRINT CHR$(27);CHR$(5);
9092 RETURN
9100 REM *** SWITCH TO SLOT 0
9101 PRINT CHR$(27);CHR$(18);
9102 RETURN
9110 REM *** SWITCH TO SLOT 1
9111 PRINT CHR$(27);CHR$(19);
9112 RETURN
9120 REM *** PROGRAM A MODE
9121 PRINT CHR$(27);CHR$(20);CHR$(8);
9122 FOR Z9=1 TO 8
```

```
9123 PRINT CHR$(M9(Z9));
9124 NEXT Z9
9125 RETURN
9130 REM *** PROGRAM JUST 1 CHARACTER GROUP
9131 PRINT CHR$(27);CHR$(20);CHR$(1);CHR$(M9(1));
9132 RETURN
9140 REM *** BLINKING CURSOR
9141 PRINT CHR$(27);CHR$(14);
9142 RETURN
9150 REM *** STEADY CURSOR
9151 PRINT CHR$(27);CHR$(13);
9153 RETURN
9160 REM *** OUTPUT TO SCREEN
9161 PRINT CHR$(27);CHR$(15);
9162 RETURN
9170 REM *** OUTPUT TO BOTH PRINTER AND SCREEN
9171 PRINT CHR$(27);CHR$(17);
9172 RETURN
9180 REM *** OUTPUT TO PRINTER ONLY
9181 PRINT CHR$(27);CHR$(16);
9182 RETURN
9190 REM *** BUILD A CHARACTER IN FONT RAM
9191 GOSUB 9080
9192 Z8=F8*16+HEX("F000")
9193 FOR Z9=1 TO 10
9194 POKE Z8,F9(Z9)
9195 Z8=Z8+1
9196 NEXT Z9
9197 GOSUB 9090
9198 RETURN
9210 REM *** OUTPUT A TONE TO THE SPEAKER
9211 POKE HEX("A03F"),D9
9212 POKE HEX("A041"),F9
9213 PRINT CHR$(07);
9214 RETURN
9999 END
```

```
8990 REM ************************************************************
8998 REM * SUPPORT PACKAGE #2 FOR GIMIX 80x24 VIDEO BOARD            *
8999 REM * CONFIGURATION: DIRECT POKES TO PORTS - NO GMXBUG          *
9000 REM ************************************************************
9001 REM
9003 REM VARIABLE USAGE:
9004 REM X9=X COORDINATE     Y9=Y COORDINATE
9005 REM M9(8) PROGRAMMING OF 1 MODE
9006 REM Z9=COUNTER  Z8=FONT ADDRESS  F9(10)=CHAR DEF
9007 REM F8=CHARACTER CODE
9008 REM D9=DURATION OF TONE  P9=PERIOD OF TONE
9020 REM ROUTINES AVAILABLE:
9021 REM 9050-SET X,Y COORDINATES
9022 REM 9060-SELECT MODE 0
9023 REM 9070-SELECT MODE 1
9024 REM 9080-SELECT FONT RAM
9025 REM 9090-SELECT REFRESH RAM
9026 REM 9100-SWITCH TO SLOT 0  (GMXBUG ONLY)
9027 REM 9110-SWITCH TO SLOT 1  (GMXBUG ONLY)
9028 REM 9120-PROGRAM A MODE
9029 REM 9130-PROGRAM JUST 1 CHARACTER GROUP
9030 REM 9140-BLINKING CURSOR
9031 REM 9150-STEADY CURSOR
9032 REM 9160-OUTPUT TO SCREEN                          (GMXBUG ONLY)
9033 REM 9170-OUTPUT TO BOTH PRINTER AND SCREEN  (GMXBUG ONLY)
9034 REM 9180-OUTPUT TO PRINTER ONLY                    (GMXBUG ONLY)
9035 REM 9190-BUILD A CHARACTER IN FONT RAM
9036 REM 9210-OUTPUT A TONE TO THE SPEAKER
9049 REM
9050 REM *** SET X-Y COORDINATES
9051 POKE HEX("F902"),X9
9052 POKE HEX("F903"),Y9
9053 Z9=(Y9-1)*80+(X9-1)+HEX("F000")
9054 POKE HEX("A018"),Z9/256
9055 POKE HEX("A019"),(Z9 AND HEX("00FF"))
9056 RETURN
9060 REM *** SELECT MODE 0
9061 Z9=PEEK(HEX("F900"))
9062 Z9=Z9 AND HEX("F7")
9063 POKE HEX("F900"),Z9
9064 RETURN
9070 REM *** SELECT MODE 1
9071 Z9=PEEK(HEX("F900"))
9072 Z9=Z9 OR HEX("08")
9073 POKE HEX("F900"),Z9
9074 RETURN
9080 REM *** SELECT FONT RAM
9081 Z9=PEEK(HEX("F900"))
9082 Z9=Z9 OR 2
9083 POKE HEX("F900"),Z9
9084 RETURN
9090 REM *** SELECT REFRESH RAM
9091 Z9=PEEK(HEX("F900"))
9092 Z9=Z9 AND HEX("FD")
```

```
9093 POKE HEX("F900"),Z9
9094 RETURN
9100 REM *** SWITCH TO SLOT 0
9102 RETURN
9110 REM *** SWITCH TO SLOT 1
9112 RETURN
9120 REM *** PROGRAM A MODE
9121 Z9=PEEK(HEX("F900")):Z9=Z9 OR 4:POKE HEX("F900"),Z9
9122 FOR Z9=1 TO 8
9123 POKE HEX("F901"),M9(Z9)
9124 NEXT Z9
9125 RETURN
9130 REM *** PROGRAM JUST 1 CHARACTER GROUP
9131 Z9=PEEK(HEX("F900"))
9132 Z9=Z9 OR 4
9133 POKE HEX("F900"),Z9
9134 POKE HEX("F901"),M9(1)
9135 RETURN
9140 REM *** BLINKING CURSOR
9141 Z9=PEEK(HEX("F900"))
9142 Z9=Z9 OR HEX("40")
9143 POKE HEX("F900"),Z9
9144 RETURN
9150 REM *** STEADY CURSOR
9151 Z9=PEEK(HEX("F900"))
9152 Z9=Z9 AND HEX("BF")
9153 POKE HEX("F900"),Z9
9154 RETURN
9160 REM *** OUTPUT TO SCREEN
9162 RETURN
9170 REM *** OUTPUT TO BOTH PRINTER AND SCREEN
9172 RETURN
9180 REM *** OUTPUT TO PRINTER ONLY
9182 RETURN
9190 REM *** BUILD A CHARACTER IN FONT RAM
9191 GOSUB 9080
9192 Z8=F8*16+HEX("F000")
9193 FOR Z9=1 TO 10
9194 POKE Z8,F9(Z9)
9195 Z8=Z8+1
9196 NEXT Z9
9197 GOSUB 9090
9198 RETURN
9210 REM *** OUTPUT A TONE TO THE SPEAKER
9211 POKE HEX("A03F"),D9
9212 POKE HEX("A041"),P9
9213 PRINT CHR$(07);
9214 RETURN
9999 END
```

MAKECHAR User's Guide

MAKECHAR is an interactive program for designing character sets for use with the Gimix 80 x 24 video board. The data for each character is displayed and may be modified with a cursor and grid system. The grid is 8 cells x 10 cells, which is the entire space of the character on the screen (there are no automatic separation cells). The cursor is indicated by a "+" or a circle. This cursor is the drawing point. Dots can be added to or removed from the pattern by moving the cursor around in DRAW or ERASE mode. The cursor can be repositioned without disturbing the pattern in MOVE mode.

MAKECHAR requires a system with GMXBUG 3.0, the Gimix 80 x 24 video board, and 4K of memory. The area for storage of a character set being edited with the program is $800-$FFF. If the user has exactly 4K, the stack must must be moved out of this area before any character set is loaded.

MAKECHAR has 21 commands, which operate as follows:

ctl-G - Moves the data in the character generator memory of the video board to MAKECHAR's buffer area.

ctl-R - Exits to GMXBUG.

0 - Sets the character to a blank space.

9 - Sets the character to a solid block.

I - Inverts each cell in the character.

+ - The grid cursor sets each cell moved into to a dot.

- - The grid cursor sets each cell moved into to blank.

(sp) - The grid cursor does not affect cells moved into.

! - The current character is saved, and the next character in numeric order is brought up for editing.

* - The current character is saved, and the previous character in numerical order is brought up for editing.

- the current character is saved, the hex number of the desired character is entered, and that character is brought up for editing.

T - The contents of MAKECHAR's character set storage area are moved to the character generator memory of the video board, and the user then can type 32 characters which will be echoed in transparent mode. "@@@" is displayed when the 32-character zone is full.

Q, W, E, A, D, Z, X, C - Moves the grid cursor in the direction indicated by the movement rosette. Movement across an edge wraps around to the other side.

ctl-L - Prints out the image of the characters being worked on. Each cell is represented by a dash for "off" or an asterisk for "on".

The user may notice one peculiarity of the "T" command. When this command is issued, the character shown in the Image position immediately changes to ASCII 00. This is because this image display is produced by putting the working character into position 0 of the character generator memory, while the T command puts the data for ASCII 00 there.

MAKECHAR loads at $100-$6E7, and the starting address is $100. The character set is stored at $800-$FFF. To save a character set, save this memory area on disk or tape.

```
                ************************************************
                *                                              *
                *       MAKECHAR Character Set Editor          *
                *                                              *
                *       For Gimix 80 x 24 Video Board          *
                *                                              *
                *       by Richard C. Rostrom                  *
                *                                              *
                *       Copyright 1979 by                      *
                *                                              *
                *       Gimix, Inc.  Chicago, Illinois         *
                *                                              *
                ************************************************


                * MODIFIED FOR 8 X 10 CHARACTER GRID  10-8-78  RCR
                * COMMAND REVISED, OTHER FIXES PUT IN  10-13-78  RCR
                * REBUILT FOR NEW VIDEO BOARD, GMXBUG 3.0  1-22-79  RCR
                * FIXED FOR VIDEO BOARD AT $F000  2-13-79  RCR
                * FIXED FOR FINAL VERSION OF GMXBUG (ESCSND CHANGED)  2-15-78
                * PRGSTR corrected for extra byte ($42)  6-11-79  rcr
                *
                * REVISED   3/14/79   RCR
                * DOT GRID USING FORM CHARACTERS, CHARACTER TYPING TEST,
                * HEX DATA DISPLAY, REFORMATTED OPTION MENU
                *
                * REVISED FOR PUBLICATION BLOCK, HARD COPY LISTING  4/3/79   R


                * STORAGE
                *
0040                    ORG     $40
0040            CHAR    RMB     1
0041            XPOS    RMB     1
0042            YPOS    RMB     1
0043            C1      RMB     1
0044            MODE    RMB     1
0045            IXX     RMB     2
0047            BITS    RMB     10
0051            XC1     RMB     1
0052            YC1     RMB     1
0053            XC2     RMB     1
0054            YC2     RMB     1
                *
0055            CHRPTR  RMB     2
0057            CHRNUM  RMB     1
0058            CHRROW  RMB     1
0059            CHRCLM  RMB     1
005A            DOTROW  RMB     1
005B            DOTCNT  RMB     1
                *
0800            CHRSET  EQU     $800
                *
                * GMXBUG VARIABLES
A002            BEGIN1  EQU     $A002
A004            END1    EQU     $A004
```

```
A014            BEGIN2  EQU     $A014
A018            CURSOR  EQU     $A018
                *
F000            SCREEN  EQU     $F000
0050            LINE    EQU     80
                *
0008            BS      EQU     8
000A            LF      EQU     $0A
000C            FF      EQU     $0C
000D            CR      EQU     $0D
001B            ESC     EQU     $1B
                *
0003            DOT     EQU     3
0004            CIRCLE  EQU     4
                *
0006            VL      EQU     6
0007            HL      EQU     7
0008            VLL     EQU     8
0009            VLR     EQU     9
000A            HLU     EQU     10
000B            HLD     EQU     11
000C            URC     EQU     12
000D            ULC     EQU     13
000E            LRC     EQU     14
000F            LLC     EQU     15
0010            HVX     EQU     16
                *
000A            CHRHGT  EQU     10
                *
0004            ADDAX   EQU     4
0005            ADDBAX  EQU     5
0008            MULTAB  EQU     8
0009            PSHALL  EQU     9
000A            PULALL  EQU     10
000E            MOVER   EQU     14
0010            INCHAR  EQU     16
0011            OUTCHR  EQU     17
0014            PSTRNG  EQU     20
0015            PSPACE  EQU     21
0016            PCRLF   EQU     22
0017            PFFEED  EQU     23
001D            PBYTE   EQU     29
001E            HEXIN   EQU     30
001F            DEBUGR  EQU     31
0028            TBSRCH  EQU     40
0029            ESCSND  EQU     41
                *
0001            POSX    EQU     1
0002            POSY    EQU     2
0005            SLCVID  EQU     5
0006            SLCFNT  EQU     6
000D            CBSTDY  EQU     13
000E            CBFLSH  EQU     14
000F            OUTSCR  EQU     15
0010            OUTPRT  EQU     16
0012            SLOT0   EQU     18
0013            SLOT1   EQU     19
0014            PRGMOD  EQU     20
0015            TRMODE  EQU     21
```

```
                        *
0100                    ORG     $100
0100 7F 00 40 START     CLR     CHAR
0103 BD 02 5A           JSR     LOADCH
0106 CE 03 A7           LDX     #PRGSTR
0109 3F                 SWI
010A 14                 FCB     PSTRNG
010B BD 02 A2 START1    JSR     DISPLY
010E CE 03 B4 MAK45     LDX     #OPTPRP
0111 3F                 SWI
0112 14                 FCB     PSTRNG
0113 3F                 SWI
0114 10                 FCB     INCHAR
0115 85 E0              BIT A   #$E0
0117 27 02              BEQ     MAK47
0119 3F                 SWI
011A 11                 FCB     OUTCHR
011B CE 01 26 MAK47     LDX     #JMPTBL
011E 3F                 SWI
011F 28                 FCB     TBSRCH
0120 26 EC              BNE     MAK45
0122 AD 00              JSR     X
0124 20 E8              BRA     MAK45
                        *
0126 57         JMPTBL  FCC     /W/
0127 01 8D              FDB     CURSUP
0129 45                 FCC     /E/
012A 01 8B              FDB     CURSUR
012C 44                 FCC     /D/
012D 01 9B              FDB     CSRRGT
012F 43                 FCC     /C/
0130 01 99              FDB     CURSDR
0132 58                 FCC     /X/
0133 01 A2              FDB     CSRDWN
0135 5A                 FCC     /Z/
0136 01 A0              FDB     CURSDL
0138 41                 FCC     /A/
0139 01 94              FDB     CSRLFT
013B 51                 FCC     /Q/
013C 01 92              FDB     CURSUL
013E 30                 FCC     /0/
013F 01 6E              FDB     BLANK
0141 39                 FCC     /9/
0142 01 6A              FDB     BLOCK
0144 49                 FCC     /I/
0145 01 7D              FDB     INVERT
0147 21                 FCC     /!/
0148 01 D5              FDB     NXTCHR
014A 2A                 FCC     /*/
014B 01 DD              FDB     PREVCH
014D 23                 FCC     /#/
014E 01 E5              FDB     SELCHR
0150 2B                 FCC     /+/
0151 02 26              FDB     DRAW
0153 2D                 FCC     /-/
0154 02 2A              FDB     ERASE
0156 20                 FCB     $20
0157 02 2D              FDB     MOVE
0159 54                 FCC     /T/
```

```
015A 01 FC              FDB     TSTCHR
015C 07                 FCB     $07
015D 03 8A              FDB     GETFNT
015F 0C                 FCB     $0C
0160 06 20              FDB     PCHRST
0162 12                 FCB     $12
0163 01 66              FDB     EXIT
0165 00                 FCB     0
                *
0166 3F         EXIT    SWI
0167 16                 FCB     PCRLF
0168 3F                 SWI
0169 1F                 FCB     DEBUGR
                *
016A C6 FF      BLOCK   LDA B   #$FF
016C 20 01              BRA     BLANKO
                *
016E 5F         BLANK   CLR B
016F 86 10      BLANKO  LDA A   #16
0171 CE 00 47           LDX     #BITS
0174 E7 00      BLANK1  STA B   X
0176 08                 INX
0177 4A                 DEC A
0178 26 FA              BNE     BLANK1
017A 7E 02 BF           JMP     DISPX
                *
017D CE 00 47   INVERT  LDX     #BITS
0180 86 0A              LDA A   #CHRHGT
0182 63 00      INV1    COM     X
0184 08                 INX
0185 4A                 DEC A
0186 26 FA              BNE     INV1
0188 7E 02 BF           JMP     DISPX
                *
018B 8D 3B      CURSUR  BSR     RIGHT
018D 8D 18      CURSUP  BSR     UP
018F 7E 02 31           JMP     DODRAW
                *
0192 8D 13      CURSUL  BSR     UP
0194 8D 28      CSRLFT  BSR     LEFT
0196 7E 02 31           JMP     DODRAW
                *
0199 8D 16      CURSDR  BSR     DOWN
019B 8D 2B      CSRRGT  BSR     RIGHT
019D 7E 02 31           JMP     DODRAW
                *
01A0 8D 1C      CURSDL  BSR     LEFT
01A2 8D 0D      CSRDWN  BSR     DOWN
01A4 7E 02 31           JMP     DODRAW
                *
                *
01A7 7A 00 42   UP      DEC     YPOS
01AA 2A 04              BPL     UP1
01AC 86 09              LDA A   #CHRHGT-1
01AE 97 42              STA A   YPOS
01B0 39         UP1     RTS
                *
01B1 7C 00 42   DOWN    INC     YPOS
01B4 96 42              LDA A   YPOS
```

```
01B6 81 0A            CMP A  #CHRHGT
01B8 26 03            BNE    DOWN1
01BA 7F 00 42         CLR    YPOS
01BD 39       DOWN1   RTS
              *
01BE 7A 00 41 LEFT    DEC    XPOS
01C1 2A 04            BPL    LEFT1
01C3 86 07            LDA A  #7
01C5 97 41            STA A  XPOS
01C7 39       LEFT1   RTS
              *
01C8 7C 00 41 RIGHT   INC    XPOS
01CB 96 41            LDA A  XPOS
01CD 81 08            CMP A  #8
01CF 26 03            BNE    RIGHT1
01D1 7F 00 41         CLR    XPOS
01D4 39       RIGHT1  RTS
              *
01D5 BD 02 7D NXTCHR  JSR    STORCH
01D8 96 40            LDA A  CHAR
01DA 4C               INC A
01DB 20 14            BRA    SELC1
              *
01DD BD 02 7D PREVCH  JSR    STORCH
01E0 96 40            LDA A  CHAR
01E2 4A               DEC A
01E3 20 0C            BRA    SELC1
              *
01E5 BD 02 7D SELCHR  JSR    STORCH
01E8 CE 06 0C         LDX    #SELMSG
01EB 3F               SWI
01EC 14               FCB    PSTRNG
01ED 3F               SWI
01EE 1E               FCB    HEXIN
01EF 29 07            BVS    NOSEL
01F1 84 7F    SELC1   AND A  #$7F
01F3 97 40            STA A  CHAR
01F5 BD 02 5A         JSR    LOADCH
01F8 BD 02 A2 NOSEL   JSR    DISPLY
01FB 39               RTS
              *
01FC BD 03 6D TSTCHR  JSR    LDFONT
01FF CE 03 BD         LDX    #CLRZON
0202 3F               SWI
0203 14               FCB    PSTRNG
0204 C6 20            LDA B  #32
0206 3F       TST1    SWI
0207 10               FCB    INCHAR
0208 36               PSH A
0209 86 15            LDA A  #TRMODE
020B 3F               SWI
020C 29               FCB    ESCSND
020D 86 01            LDA A  #1
020F 3F               SWI
0210 11               FCB    OUTCHR
0211 32               PUL A
0212 3F               SWI
0213 11               FCB    OUTCHR
0214 5A               DEC B
```

```
0215 26 EF            BNE     TST1
0217 CE 03 E9         LDX     #XTSTR
021A 3F               SWI
021B 14               FCB     PSTRNG
021C 3F               SWI
021D 10               FCB     INCHAR
021E CE 03 A7         LDX     #PRGSTR
0221 3F               SWI
0222 14               FCB     PSTRNG
0223 7E 02 A2         JMP     DISPLY
                      *
0226 86 01    DRAW    LDA A   #1
0228 20 05            BRA     SETMOD
                      *
022A 4F       ERASE   CLR A
022B 20 02            BRA     SETMOD
                      *
022D 86 FF    MOVE    LDA A   #$FF
022F 97 44    SETMOD  STA A   MODE
                      *
0231 CE 00 47 DODRAW  LDX     #BITS
0234 96 42            LDA A   YPOS
0236 27 04            BEQ     DODR2
0238 08       DODR1   INX
0239 4A               DEC A
023A 26 FC            BNE     DODR1
023C C6 80    DODR2   LDA B   #%10000000
023E 96 41            LDA A   XPOS
0240 27 04            BEQ     DODR4
0242 54       DODR3   LSR B
0243 4A               DEC A
0244 26 FC            BNE     DODR3
0246 96 44    DODR4   LDA A   MODE
0248 2B 0D            BMI     NODRAW
024A 26 07            BNE     SETON
024C 53               COM B
024D E4 00            AND B   X
024F E7 00            STA B   X
0251 20 04            BRA     NODRAW
0253 EA 00    SETON   ORA B   X
0255 E7 00            STA B   X
0257 7E 02 BF NODRAW  JMP     DISPX
                      *
025A BD 02 94 LOADCH  JSR     FNDENT
025D E6 00    LOAD1   LDA B   X
025F 37               PSH B
0260 08               INX
0261 4A               DEC A
0262 26 F9            BNE     LOAD1
0264 86 0A            LDA A   #CHRHGT
0266 CE 00 50         LDX     #BITS+CHRHGT-1
0269 33       LOAD2   PUL B
026A E7 00            STA B   X
026C 09               DEX
026D 4A               DEC A
026E 26 F9            BNE     LOAD2
0270 86 04            LDA A   #4
0272 97 41            STA A   XPOS
0274 86 08            LDA A   #8
```

```
0276 97 42          STA A  YPOS
0278 86 FF          LDA A  #$FF
027A 97 44          STA A  MODE
027C 39             RTS
                    *
027D 86 0A    STORCH LDA A  #CHRHGT
027F CE 00 50       LDX    #BITS+CHRHGT-1
0282 E6 00    STOR1 LDA B   X
0284 37             PSH B
0285 09             DEX
0286 4A             DEC A
0287 26 F9          BNE    STOR1
0289 BD 02 94       JSR    FNDENT
028C 33       STOR2 PUL B
028D E7 00          STA B  X
028F 08             INX
0290 4A             DEC A
0291 26 F9          BNE    STOR2
0293 39             RTS
                    *
0294 CE 08 00 FNDENT LDX   #CHRSET
0297 96 40          LDA A  CHAR
0299 C6 10          LDA B  #16
029B 3F             SWI
029C 08             FCB    MULTAB
029D 3F             SWI
029E 05             FCB    ADDBAX
029F 86 0A          LDA A  #CHRHGT
02A1 39             RTS
                    *
02A2 CE 03 EF DISPLY LDX   #PTRSTR
02A5 3F       DISPY1 SWI
02A6 09             FCB    PSHALL
02A7 EE 00          LDX    X
02A9 3F             SWI
02AA 14             FCB    PSTRNG
02AB 3F             SWI
02AC 0A             FCB    PULALL
02AD 08             INX
02AE 08             INX
02AF 8C 04 2F       CPX    #XPS
02B2 26 F1          BNE    DISPY1
02B4 86 10          LDA A  #16
02B6 C6 11          LDA B  #17
02B8 BD 03 3B       JSR    SETXY
02BB 96 40          LDA A  CHAR
02BD 3F             SWI
02BE 1D             FCB    PBYTE
02BF BD 03 4C DISPX JSR    DOFONT
02C2 86 0A          LDA A  #10
02C4 C6 13          LDA B  #19
02C6 8D 73          BSR    SETXY
02C8 CE 00 47       LDX    #BITS
02CB A6 00    DISPX1 LDA A  X
02CD 3F             SWI
02CE 1D             FCB    PBYTE
02CF 3F             SWI
02D0 15             FCB    PSPACE
02D1 08             INX
```

```
02D2 8C 00 51          CPX     #BITS+CHRHGT
02D5 26 F4             BNE     DISPX1
02D7 CE 00 47          LDX     #BITS
02DA DF 45             STX     IXX
02DC 86 0A             LDA A   #CHRHGT
02DE 97 52             STA A   YC1
02E0 7F 00 54          CLR     YC2
02E3 86 08     DISP1   LDA A   #8
02E5 97 51             STA A   XC1
02E7 7F 00 53          CLR     XC2
02EA DE 45             LDX     IXX
02EC A6 00             LDA A   X
02EE 97 43             STA A   C1
02F0 D6 54             LDA B   YC2
02F2 58                ASL B
02F3 5C                INC B
02F4 8D 47             BSR     SETY
02F6 86 20     DISP2   LDA A   #$20
02F8 78 00 43          ASL     C1
02FB 24 02             BCC     DISP25
02FD 86 03             LDA A   #DOT
02FF 36        DISP25  PSH A
0300 96 53             LDA A   XC2
0302 48                ASL A
0303 8B 3D             ADD A   #61
0305 8D 3C             BSR     SETX
0307 FE A0 18          LDX     CURSOR
030A 32                PUL A
030B A7 00             STA A   X
030D 7C 00 53  DISP3   INC     XC2
0310 7A 00 51          DEC     XC1
0313 26 E1             BNE     DISP2
0315 7C 00 46          INC     IXX+1
0318 7C 00 54          INC     YC2
031B 7A 00 52          DEC     YC1
031E 26 C3             BNE     DISP1
0320 96 41             LDA A   XPOS
0322 D6 42             LDA B   YPOS
0324 48                ASL A
0325 58                ASL B
0326 8B 3D             ADD A   #61
0328 5C                INC B
0329 8D 10             BSR     SETXY
032B FE A0 18          LDX     CURSOR
032E C6 04             LDA B   #CIRCLE
0330 A6 00             LDA A   X
0332 81 03             CMP A   #DOT
0334 27 02             BEQ     DISP4
0336 C6 2B             LDA B   #'+
0338 E7 00     DISP4   STA B   X
033A 39                RTS
               *
033B 8D 06     SETXY   BSR     SETX
033D 17        SETY    TBA
033E 36                PSH A
033F 86 02             LDA A   #POSY
0341 20 03             BRA     SETY1
0343 36        SETX    PSH A
0344 86 01             LDA A   #POSX
```

```
0346 3F          SETY1   SWI
0347 29                  FCB     ESCSND
0348 32                  PUL A
0349 3F                  SWI
034A 11                  FCB     OUTCHR
034B 39                  RTS
                 *
                 * COPY CURRENT CHAR TO FONT RAM POSITION 1
034C 86 06       DOFONT  LDA A   #SLCFNT
034E 3F                  SWI
034F 29                  FCB     ESCSND
0350 CE 00 50            LDX     #BITS+CHRHGT-1
0353 C6 0A               LDA B   #CHRHGT
0355 A6 00       DOF1    LDA A   X
0357 36                  PSH A
0358 09                  DEX
0359 5A                  DEC B
035A 26 F9               BNE     DOF1
035C CE F0 00            LDX     #SCREEN
035F C6 0A               LDA B   #CHRHGT
0361 32          DOF2    PUL A
0362 A7 00               STA A   X
0364 08                  INX
0365 5A                  DEC B
0366 26 F9               BNE     DOF2
0368 86 05               LDA A   #SLCVID
036A 3F                  SWI
036B 29                  FCB     ESCSND
036C 39                  RTS
                 *
                 * LOAD FONT RAM FROM WORKING AREA
036D 86 06       LDFONT  LDA A   #SLCFNT
036F 3F                  SWI
0370 29                  FCB     ESCSND
0371 CE 08 00            LDX     #CHRSET
0374 FF A0 02            STX     BEGIN1
0377 CE 0F FF            LDX     #CHRSET+2047
037A FF A0 04            STX     END1
037D CE F0 00            LDX     #SCREEN
0380 FF A0 14            STX     BEGIN2
0383 3F                  SWI
0384 0E                  FCB     MOVER
0385 86 05               LDA A   #SLCVID
0387 3F                  SWI
0388 29                  FCB     ESCSND
0389 39                  RTS
                 *
038A 86 06       GETFNT  LDA A   #SLCFNT
038C 3F                  SWI
038D 29                  FCB     ESCSND
038E CE F0 03            LDX     #SCREEN+3
0391 FF A0 02            STX     BEGIN1
0394 CE F7 FF            LDX     #SCREEN+2047
0397 FF A0 04            STX     END1
039A CE 08 03            LDX     #CHRSET+3
039D FF A0 14            STX     BEGIN2
03A0 3F                  SWI
03A1 0E                  FCB     MOVER
03A2 86 05               LDA A   #SLCVID
```

```
03A4 3F                 SWI
03A5 29                 FCB     ESCSND
03A6 39                 RTS
                *
03A7 1B         PRGSTR  FCB     ESC,PRGMOD,9,$42,0,$10,$20,$30
03AF 42                 FCB     $42,$52,$62,$72,4
                *
03B4 1B         OPTPRP  FCB     ESC,POSY,21,ESC,POSX,11,$20,BS,4
                *
03BD 1B         CLRZON  FCB     ESC,POSX,15,ESC,2,21
03C3 20                 FCC     /                                       /
03E3 1B                 FCB     ESC,POSX,15,ESC,SLOT1,4
                *
03E9 1B         XTSTR   FCB     ESC,SLOTO
03EB 40                 FCC     /@@@/
03EE 04                 FCB     4
                *
03EF 04 2F      PTRSTR  FDB     GRD1,MENU1,GRD2,GRD3,MENU2,GRD2,GRD3,MENU3,GRD2
0401 04 64              FDB     GRD3,MENU4,GRD2,GRD3,MENU5,GRD2,GRD3,MENU6,GRD2
0413 04 64              FDB     GRD3,MENU7,GRD2,GRD3,MENU8,GRD2,GRD3,MENU9,GRD2
0425 04 64              FDB     GRD3,MENU10,GRD2,GRD4,MENU11
042F            XPS     EQU     *
                *
042F 0C         GRD1    FCB     FF,ESC,POSX,60,ESC,TRMODE,17
0436 0D                 FCB     ULC,HL,HLD,HL,HLD,HL,HLD,HL,HLD,HL,HLD,HL
0442 0B                 FCB     HLD,HL,HLD,HL,URC
0447 0D                 FCB     $D,$A,4
                *
044A 1B         GRD2    FCB     ESC,POSX,60,ESC,TRMODE,17
0450 06                 FCB     VL,$20,VL,$20,VL,$20,VL,$20,VL,$20,VL,$20
045C 06                 FCB     VL,$20,VL,$20,VL
0461 0D                 FCB     $D,$A,4
                *
0464 1B         GRD3    FCB     ESC,POSX,60,ESC,TRMODE,17
046A 09                 FCB     VLR,HL,HVX,HL,HVX,HL,HVX,HL,HVX,HL,HVX,HL
0476 10                 FCB     HVX,HL,HVX,HL,VLL
047B 0D                 FCB     $D,$A,4
                *
047E 1B         GRD4    FCB     ESC,POSX,60,ESC,TRMODE,17
0484 0F                 FCB     LLC,HL,HLU,HL,HLU,HL,HLU,HL,HLU,HL,HLU,HL
0490 0A                 FCB     HLU,HL,HLU,HL,LRC
0495 0D                 FCB     $D,$A,4
                *
0498 5E         MENU1   FCC     /^G to set font char set  ^R to return to GMXBL
04C7 04                 FCB     4
04C8 30         MENU2   FCC     /0 for BLANK    9 for BLOCK     I for INVERSE/
04F3 04                 FCB     4
04F4 2B         MENU3   FCC     /+ to DRAW      - to ERASE    (sp)  to MOVE/
051E 04                 FCB     4
051F 21         MENU4   FCC     /! for NEXT CODE      * for LAST CODE/
0542 04                 FCB     4
0543 23         MENU5   FCC     /# to PICK A CODE     T to TEST CHAR SET/
0569 04                 FCB     4
056A 5E         MENU6   FCC     /^L for listing                  Q  W  E/
058E 04                 FCB     4
058F 20         MENU7   FCC     /                Movement:      A  *  D/
05B3 04                 FCB     4
05B4 20         MENU8   FCC     /                               Z  X  C/
05D8 04                 FCB     4
```

```
05D9 43          MENU9   FCC     /Character code:       Image: /
05F6 80                  FCB     $80,4
05F8 48          MENU10  FCC     /Hex data: /
0602 04                  FCB     4
0603 4F          MENU11  FCC     /Option? /
060B 04                  FCB     4
060C 0D          SELMSG  FCB     $D,$A
060E 45                  FCC     /Enter char code: /
061F 04                  FCB     4
                 *
                 * PRINT OUT CHARACTER SET ON THE SYSTEM HARDCOPY DEVICE
0620 86 10       PCHRST  LDA A   #OUTPRT
0622 3F                  SWI
0623 29                  FCB     ESCSND
                 *
0624 7F 00 57            CLR     CHRNUM
0627 CE 08 00            LDX     #CHRSET
062A DF 55               STX     CHRPTR
062C 3F          NEWPGE  SWI
062D 16                  FCB     PCRLF
062E 86 05               LDA A   #5
0630 97 58               STA A   CHRROW
0632 86 0A       NEWROW  LDA A   #10
0634 97 5A               STA A   DOTROW
0636 86 05       NCHRRW  LDA A   #5
0638 97 59               STA A   CHRCLM
063A CE 06 E1            LDX     #SPSTRG
063D 3F                  SWI
063E 14                  FCB     PSTRNG
063F 7D 00 57    NEWCLM  TST     CHRNUM
0642 2B 5B               BMI     SKPDOT
0644 96 5A               LDA A   DOTROW
0646 81 0A               CMP A   #10
0648 26 14               BNE     NEWC1
064A 96 57               LDA A   CHRNUM
064C 85 60               BIT A   #$60
064E 27 24               BEQ     NEWC2
0650 81 7F               CMP A   #$7F
0652 27 20               BEQ     NEWC2
0654 B7 06 DD            STA A   NAMBYT
0657 CE 06 DA            LDX     #NAMSTR
065A 3F                  SWI
065B 14                  FCB     PSTRNG
065C 20 1B               BRA     NEWC3
                 *
                 * PRINT HEX OF CHARACTER
065E 81 09       NEWC1   CMP A   #9
0660 26 12               BNE     NEWC2
0662 3F                  SWI
0663 15                  FCB     PSPACE
0664 86 24               LDA A   #'$
0666 3F                  SWI
0667 11                  FCB     OUTCHR
0668 96 57               LDA A   CHRNUM
066A 3F                  SWI
066B 1D                  FCB     PBYTE
066C 86 3A               LDA A   #':
066E 3F                  SWI
066F 11                  FCB     OUTCHR
```

```
0670 3F                 SWI
0671 15                 FCB     PSPACE
0672 20 05              BRA     NEWC3
                *
                * PRINT 6 BLANKS
0674 CE 06 E1   NEWC2   LDX     #SPSTRG
0677 3F                 SWI
0678 14                 FCB     PSTRNG
                *
                * PRINT 1 CHARACTER'S DOTS
0679 DE 55      NEWC3   LDX     CHRPTR
067B 86 05              LDA A   #5
067D 90 59              SUB A   CHRCLM
067F 48                 ASL A
0680 48                 ASL A
0681 48                 ASL A
0682 48                 ASL A
0683 3F                 SWI
0684 04                 FCB     ADDAX
0685 86 0A              LDA A   #10
0687 90 5A              SUB A   DOTROW
0689 3F                 SWI
068A 04                 FCB     ADDAX
068B E6 00              LDA B   X
068D 86 08              LDA A   #8
068F 97 5B              STA A   DOTCNT
0691 86 2D      DOT1    LDA A   #'-
0693 58                 ASL B
0694 24 02              BCC     DOT2
0696 86 2A              LDA A   #'*
0698 3F         DOT2    SWI
0699 11                 FCB     OUTCHR
069A 7A 00 5B           DEC     DOTCNT
069D 26 F2              BNE     DOT1
                *
                * PRINT SAME ROW OF NEXT CHARACTER
069F 7C 00 57   SKPDOT  INC     CHRNUM
06A2 7A 00 59           DEC     CHRCLM
06A5 26 98              BNE     NEWCLM
                *
                * PRINT NEXT DOT ROW OF THE FIVE CHARACTERS
06A7 3F                 SWI
06A8 16                 FCB     PCRLF
06A9 96 57              LDA A   CHRNUM
06AB 80 05              SUB A   #5
06AD 97 57              STA A   CHRNUM
06AF 7A 00 5A           DEC     DOTROW
06B2 26 82              BNE     NCHRRW
                *
                * START A NEW ROW OF CHARACTERS
06B4 3F                 SWI
06B5 16                 FCB     PCRLF
06B6 3F                 SWI
06B7 16                 FCB     PCRLF
06B8 3F                 SWI
06B9 16                 FCB     PCRLF
06BA DE 55              LDX     CHRPTR
06BC 86 50              LDA A   #80
06BE 3F                 SWI
```

```
06BF 04                    FCB    ADDAX
06C0 DF 55                 STX    CHRPTR
06C2 96 57                 LDA A  CHRNUM
06C4 8B 05                 ADD A  #5
06C6 2B 0D                 BMI    DONE
06C8 97 57                 STA A  CHRNUM
06CA 7A 00 58              DEC    CHRROW
06CD 27 03                 BEQ    NXTPAG
06CF 7E 06 32              JMP    NEWROW
06D2 7E 06 2C  NXTPAG      JMP    NEWPGE
               *
               *
06D5 86 0F     DONE        LDA A  #OUTSCR
06D7 3F                    SWI
06D8 29                    FCB    ESCSND
06D9 39                    RTS
               *
               *
06DA 20        NAMSTR      FCC    /  [/
06DD           NAMBYT      RMB    1
06DE 5D                    FCC    /] /
06E0 04                    FCB    4
06E1 20        SPSTRG      FCC    /      /
06E7 04                    FCB    4
                           END
```

NO ERROR(S) DETECTED

SYMBOL TABLE:

ADDAX	0004	ADDBAX	0005	BEGIN1	A002	BEGIN2	A014	BITS	0047
BLANK	016E	BLANK0	016F	BLANK1	0174	BLOCK	016A	BS	0008
C1	0043	CBFLSH	000E	CBSTDY	000D	CHAR	0040	CHRCLM	0059
CHRHGT	000A	CHRNUM	0057	CHRPTR	0055	CHRROW	0058	CHRSET	0800
CIRCLE	0004	CLRZON	03BD	CR	000D	CSRDWN	01A2	CSRLFT	0194
CSRRGT	019B	CURSDL	01A0	CURSDR	0199	CURSOR	A018	CURSUL	0192
CURSUP	018D	CURSUR	018B	DEBUGR	001F	DISP1	02E3	DISP2	02F6
DISP25	02FF	DISP3	030D	DISP4	0338	DISPLY	02A2	DISPX	02BF
DISPX1	02CB	DISPY1	02A5	DODR1	0238	DODR2	023C	DODR3	0242
DODR4	0246	DODRAW	0231	DOF1	0355	DOF2	0361	DOFONT	034C
DONE	06D5	DOT	0003	DOT1	0691	DOT2	0698	DOTCNT	005B
DOTROW	005A	DOWN	01B1	DOWN1	01BD	DRAW	0226	END1	A004
ERASE	022A	ESC	001B	ESCSND	0029	EXIT	0166	FF	000C
FNDENT	0294	GETFNT	038A	GRD1	042F	GRD2	044A	GRD3	0464
GRD4	047E	HEXIN	001E	HL	0007	HLD	000B	HLU	000A
HVX	0010	INCHAR	0010	INV1	0182	INVERT	017D	IXX	0045
JMPTBL	0126	LDFONT	036D	LEFT	01BE	LEFT1	01C7	LF	000A
LINE	0050	LLC	000F	LOAD1	025D	LOAD2	0269	LOADCH	025A
LRC	000E	MAK45	010E	MAK47	011B	MENU1	0498	MENU10	05F8
MENU11	0603	MENU2	04C8	MENU3	04F4	MENU4	051F	MENU5	0543
MENU6	056A	MENU7	058F	MENU8	05B4	MENU9	05D9	MODE	0044
MOVE	022D	MOVER	000E	MULTAB	0008	NAMBYT	06DD	NAMSTR	06DA
NCHRRW	0636	NEWC1	065E	NEWC2	0674	NEWC3	0679	NEWCLM	063F
NEWPGE	062C	NEWROW	0632	NODRAW	0257	NOSEL	01F8	NXTCHR	01D5
NXTPAG	06D2	OPTPRP	03B4	OUTCHR	0011	OUTPRT	0010	OUTSCR	000F
PBYTE	001D	PCHRST	0620	PCRLF	0016	PFFEED	0017	POSX	0001
POSY	0002	PREVCH	01DD	PRGMOD	0014	PRGSTR	03A7	PSHALL	0009
PSPACE	0015	PSTRNG	0014	PTRSTR	03EF	PULALL	000A	RIGHT	01C8
RIGHT1	01D4	SCREEN	F000	SELC1	01F1	SELCHR	01E5	SELMSG	060C

SETMOD	022F	SETON	0253	SETX	0343	SETXY	033B	SETY	033D
SETY1	0346	SKPDOT	069F	SLCFNT	0006	SLCVID	0005	SLOT0	0012
SLOT1	0013	SPSTRG	06E1	START	0100	START1	010B	STOR1	0282
STOR2	028C	STORCH	027D	TBSRCH	0028	TRMODE	0015	TST1	0206
TSTCHR	01FC	ULC	000D	UP	01A7	UP1	01B0	URC	000C
VL	0006	VLL	0008	VLR	0009	XC1	0051	XC2	0053
XPOS	0041	XPS	042F	XTSTR	03E9	YC1	0052	YC2	0054
YPOS	0042								